

파일 보안 서버 프로그램

(File Security Server Program)

팀 명 : TEAM S.F
지도 교수 : 이병천 교수님
팀 장 : 윤준원
팀 원 : 박태형
정호윤
이희웅
정세욱

목 차

1. 서론

2. 관련연구

2.1 API

2.2 Socket

2.3 hight

2.4 RSA

3. 본론

3.1 API - GUI환경의 클라이언트

3.2 Socket 프로그래밍 - 파일 송수신 기능

3.3 hight와 RSA알고리즘 - 암호화 기능

3.4 주요 기능 코드

4. 결론

5. 참고 자료

6. 발표 자료

1. 서론

우리나라는 IT분야에서 빠른 발전과 높은 성장률을 이루었다. 그에 따른 부작용으로 발전에 비례하지 않게 보안기능이 결여되어 있다. 인터넷이 보급된 이후로 지금까지 파일 유출에 대한 문제는 지속적으로 제기되어 왔다.

초기 인터넷환경에서 지금까지 사용되고 있는 FTP나 Telnet, putty등 다양한 파일 송수신 프로그램이 있지만 전송하려는 파일에 대한 보안이 취약하다는 공통점을 가지고 있다.

그렇기에 대부분의 기업에서 많은 돈을 들여 정보보안에 힘을 쓰고 있다. 하지만 학교나 중소기업과 같은 소규모 네트워크 환경을 구축하고 있는 곳에서는 이러한 노력이 부족하고 보안의식 또한 떨어지기 때문에 다양한 유출사고가 빈번하게 일어나고 있다.

Team S.F.에서는 그에 대한 문제점을 해결하기 위해 소규모 네트환경의 파일보안을 이번 프로젝트로 진행하게 되었다.

2. 관련연구

2.1 API

API란 Application Programming Interface의 약자이며 운영체제와 응용프로그램 사이의 통신에 사용되는 언어나 메시지 형식을 말한다.

API는 소프트웨어 컴포넌트의 기능, 입력, 출력, 그리고 이에 사용되는 자료형으로 표현된다. API 자체는 어디까지나 사양(Specification)만을 정의하기 때문에 구현(Implementation)물과 독립적이다. API는 다양한 형태로 존재하며, 유닉스의 POSIX 표준, 윈도우의 MFC나 Win32, C++의 Standard Template Library(STL), Java API 등이 이에 해당한다.

API가 프로그래머를 위해서 만들어지기는 했지만, 사용자 입장에서든 같은 API를 사용한 프로그램은 비슷한 인터페이스를 가지기 때문에 새로운 프로그램의 사용법을 배우기가 쉬워진다.

예를 들어 그래픽 카드나 디스크 드라이브 등의 하드웨어 또는 데이터베이스를 조작할 때, API는 작업을 편리하게 해준다. API는 이러한 작업들에 대한 기능을 대상이 되는 언어에 맞게 추상화하고 프로그래머가 사용하기 편리하게 설계된 인터페이스 사양이다.

프로그램에 플러그인 형태로 설계된 API가 적용되면 이미 작성되고 컴파일까지 끝난 프로그램이라도 수정 없이 프로그램의 기능을 추가하는 것이 가능하다. 인터넷 익스플로러, 파이어폭스, 크롬과 같은 웹 브라우저 프로그램의 플러그인, 애드온과 같은 것이 바로 이러한 형식의 플러그인 API를 사용해 구현된 것이다.

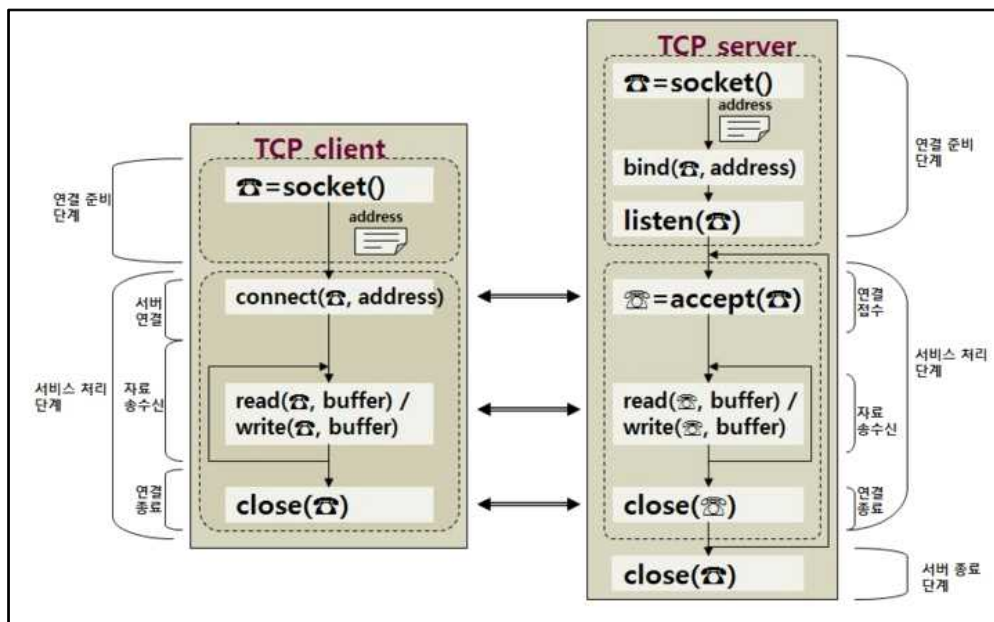
API가 실제 기능 구현체인 라이브러리와 함께 제공되는 경우도 있으며, 이 경우를 보통 SDK(Software Development Kit)라고 한다. SDK는 보통 API, 라이브러리와 함께 프로그램을 개발하는데 필요한 여러 보조 프로그램이 같이 배포된다.

라이브러리는 실제로 동작할 수 있는 단편화된 프로그램이라는 점에서 API와 다르다. 라이브러리 자체는 API 없이 존재할 수 있고, 이미 구현되어 기계어로 컴파일된 프로그램에 의해 사용될 수도 있다.

API 없이 프로그램을 실행하는데 필요한 라이브러리만 배포되는 대표적인 경우로 "Visual C++ Runtime Library", "DirectX Runtime"이 있다.

2.2 Socket

소켓 네트워크 프로그래밍이란 소켓 함수를 이용, 소켓 객체를 만들어서 인터넷을 가로 질러서 정보를 교환할 수 있는 소프트웨어를 만드는 프로그래밍 기술을 말하며 서버/클라이언트 모델에서 프로그램은 쌍은 개발된다.



<그림1. TCP 소켓프로그래밍 통신 단계>

전체적인 기능 수행 방식은 TCP/IP 통신망에 연결된 클라이언트가 서버에서 실행 중인 응용프로그램에 연결할 수 있도록 접속점을 제공하며 연결 요청을 해당 서버의 응용프로그램으로 전달된다. 연결 요청을 받아들이면 클라이언트와 연결하여 약속된 규약에 따라 자료를 송수신한다.

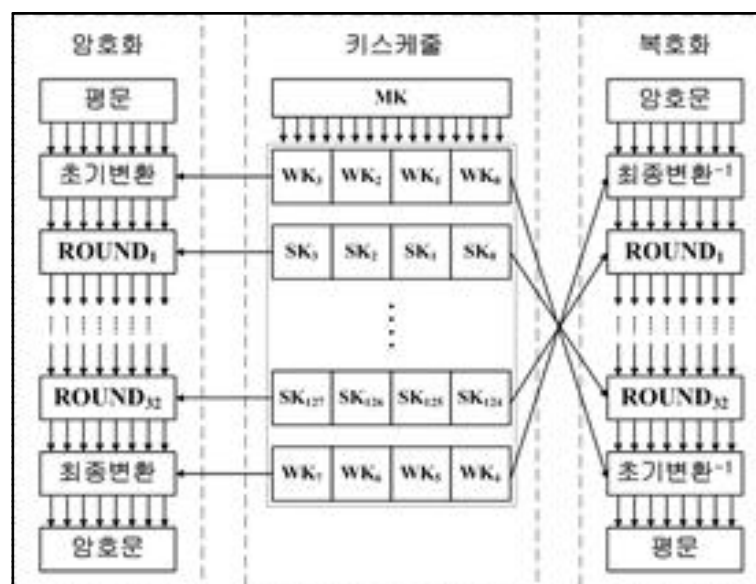
자료 송수신이 더 이상 필요 없다면 서비스 처리를 담당했던 해당 연결 소켓을 종료해준다.

2.3 hight

국내 스마트폰 보급률이 증가하면서 언제 어디서나 자유롭게 네트워크에 접속할 수 있는 유비쿼터스화가 이루어졌다. 그러나 이러한 환경에서 데이터를 보호하기

위한 암호기술이 적용되지 않은 채로 개인정보, 공문서와 같은 중요데이터들의 정보교환이 이루어지고 있으며, 스마트폰 등의 모바일 기기가 PC에 비하여 타인이 쉽게 접근 가능하다는 점과 기기 분실의 위험이 크다는 점을 고려, 모바일 기기의 안정성·신뢰성 제고가 절실하다. 이를 위한 가장 기본적인 방법이 암호화 기술 적용이다.

최근 암호는 RFID, USN, 스마트폰 등과 같이 초고속·초경량·저전력을 요구하는 유비쿼터스 컴퓨팅 환경에 적용될 수 있도록 축소되고 있다. 여기서 제안하는 HIGHT(HIGH security and light weight)는 초경량블록암호 알고리즘으로써 순수 우리기술로 개발, 국제표준화기구(ISO/IEC)에 표준으로 채택되었고 한국인터넷진흥원(KISA)의 주요사업으로 암호기술의 이용을 촉진하고 있다.

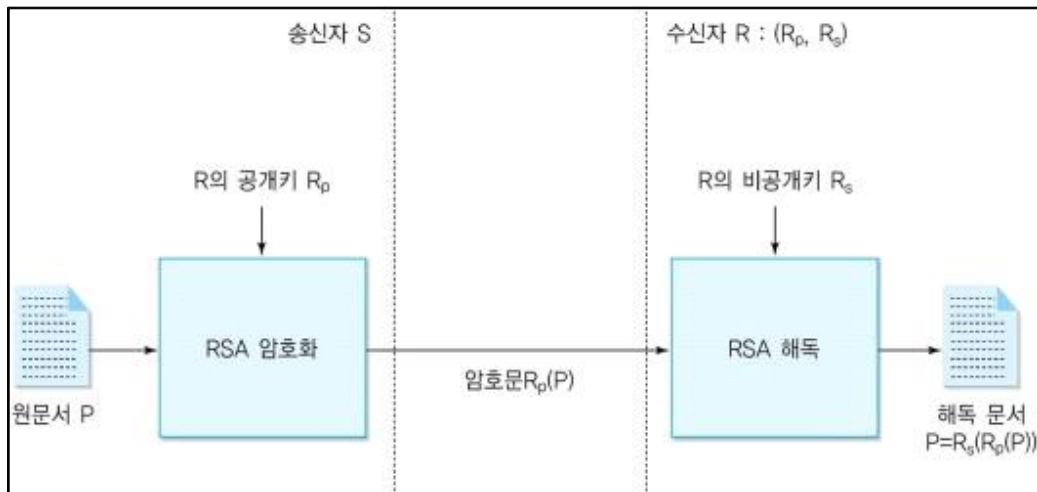


<그림2. HIGHT 암호 알고리즘>

2.4 RSA

RSA 암호는 리베스트(Rivest, R.), 샤미르(Shamir, A.), 애들먼(Adleman, L.)이 1977년에 개발한 암호 체계(crypto system)로, 처음으로 상용화되었고 지금도 널리 쓰이는 대표적인 공개키 암호 체계이다.

RSA의 안전성은 매우 큰 정수의 소인수분해가 어렵다는 점에 기반하고 있다. 이 방식은 두 개의 큰 소수들의 곱과 추가 연산을 통해 공개키와 비밀키를 구한다. 이 과정을 거쳐 키들이 생성되면 처음의 두 소수는 더 이상 중요하지 않고 버려도 무방하다. 공개키는 모두에게 공개되는 열쇠지만, 비밀키는 자신만이 가지고 있는 열쇠이므로 남에게 공개되어서는 안 된다. 이런 비밀키는 공개키에 의해 암호화된 메시지를 복호화 하는데 사용된다.



<그림3. RSA 암호 알고리즘>

3. 본론

3.1 API - GUI환경의 클라이언트

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)
{
    INITCOMMONCONTROLSEX InitCtrls;
    InitCtrls.dwICC = ICC_LISTVIEW_CLASSES;
    InitCtrls.dwSize = sizeof(INITCOMMONCONTROLSEX);
    BOOL bRet = InitCommonControlsEx(&InitCtrls);

    hInst=hInstance;

    DialogBoxParam(hInstance, MAKEINTRESOURCE(IDC_DIALOG), NULL, (DLGPROC)DialogProc,0);
    return 0;
}
```

- 기본 다이얼 로그를 생성한다. 생성된 다이얼 로그 박스는 아무것도 들어있지 않다.

```
case WM_INITDIALOG:
{
    char Temp[255];
    LVBKIMAGE plvbki={0};
    LVBKIMAGE ki={0};
    char url[]="C:\\a.jpg";
    InitCommonControls();
    hList=GetDlgItem(hWnd,IDC_LIST);

    memset(&plvbki,0,sizeof(plvbki));
    plvbki.ulFlags=LVBKIF_SOURCE_URL;
    plvbki.pszImage=url;
    plvbki.xOffsetPercent=40;
    plvbki.yOffsetPercent=15;
    OleInitialize(NULL);

    SendMessage(hList,LVM_SETTEXTBKCOLOR, 0,(LPARAM)CLR_NONE);
    SendMessage(hList,LVM_SETBKIMAGE,0,(LPARAM)(LPLVBKIMAGE)&plvbki);
    SendMessage(hList,LVM_SETEXTENDEDLISTVIEWSTYLE,0,LVS_EX_FULLROWSELECT);
    memset(&LvCol,0,sizeof(LvCol)); // Reset Coluom
    LvCol.mask=LVCF_TEXT|LVCF_WIDTH|LVCF_SUBITEM; // Type of mask

    LvCol.cx=0x60;
    LvCol.pszText="파일 이름";
    SendMessage(hList,LVM_INSERTCOLUMN,0,(LPARAM)&LvCol);
    LvCol.cx=0x42;
    LvCol.pszText="파일 유형";
    SendMessage(hList,LVM_INSERTCOLUMN,1,(LPARAM)&LvCol);
    LvCol.pszText="파일 크기";
    SendMessage(hList,LVM_INSERTCOLUMN,2,(LPARAM)&LvCol);
}
```

- 다이얼 로그 초기화면을 세팅해주는 소스이다. 리스트 박스의 기본 초기 화면 스타일 세팅 및 칼럼 부분을 설정해준다.

```

BOOL CALLBACK DialogProc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)
{
    switch(Message)
    {
        HDC hdc, MemDC;
        PAINTSTRUCT ps;
        HBITMAP MyBitmap, OldBitmap;

        case WM_CLOSE:
        {
            PostQuitMessage(0);
            EndDialog(hWnd,0); // 다이얼 로그 끝내기
        }
        break;

        case WM_NOTIFY:
        {
            switch(((LPNMHDR)lParam)->code)
            {
                case NM_DBLCLK: //더블클릭을 했을 때
                if (((LPNMHDR)lParam)->idFrom == IDC_LIST)
                {
                    char Text[255]={0};
                    char Temp[255]={0};
                    char Temp1[255]={0};

```

- 더블클릭, 엔터 등 각각의 요소 안에 코드를 집어넣어 파일 암호화 및 전송을 해주는 코드를 넣는다.

```

case WM_PAINT:
{
    hdc = BeginPaint(hWnd, &ps);
    MemDC=CreateCompatibleDC(hdc);
    MyBitmap=LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP4));
    OldBitmap=(HBITMAP)SelectObject(MemDC, MyBitmap);
    BitBlt(hdc, 300, 80, 250, 250,MemDC,0,0,SRCCOPY);
    SelectObject(MemDC,OldBitmap);
    DeleteObject(MyBitmap);
    DeleteDC(MemDC);
}

```

- 프로그램 상에서 팀 로고, 학교 마크, 사진 등을 첨부하는 기능이다.

```

#define IDC_DIALOG          101
#define IDI_ICON1          102
#define IDR_MENU1          106
#define IDI_ICON2          116
#define IDD_DIALOG1        121
#define IDB_BITMAP1        123
#define IDC_LIST            1000
#define IDC_LIST2          1003
#define IDC_COMBOBOXEX1    1015
#define IDC_COMBO1         1016
#define IDC_RENAME         40002
#define IDC_SELECT_ALL     40004
#define IDC_LAST_ITEM     40005

#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        124
#define _APS_NEXT_COMMAND_VALUE        40006
#define _APS_NEXT_CONTROL_VALUE        1017
#define _APS_NEXT_SYMED_VALUE          101
#endif
#endif

```

- 다이얼로그, 비트맵, 리스트 박스, 콤보 박스 등 resource.h에 정의되어 있는 자원들을 사용할 수 있다.

3.2 Socket 프로그래밍 - 파일 송수신 기능

```

char *result;
WSADATA wsaData;
int retval = WSASStartup(MAKEWORD(2,2),&wsaData);

if(retval != 0)
{
    printf("WSAStartup() Error\n");
    return 0;
}

SOCKET serv_sock = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
SOCKADDR_IN serv_addr;
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(4000);
serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

retval = connect(serv_sock,(SOCKADDR*)&serv_addr,sizeof(SOCKADDR));
if(retval == SOCKET_ERROR)
{
    printf("connect() Error\n");
    return 0;
}
char buf[256];
char buf2[1024]={0};

strcpy(buf, "2:");
strcat(buf, str);
strcat(buf, ":");

int sendsize = send(serv_sock,buf,strlen(buf),0);
if(sendsize <= 0)
break;

int recvsize = recv(serv_sock,buf2,sizeof(buf2),0);
if(recvsize <= 0)

```

- connect 함수를 통해 현재 서버에 접속하는 프로그래밍

```

//-----서버(자신)의 소켓 정보 입력-----
SOCKADDR_IN serv_addr = {0};
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(4000);
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
//-----

//-----인터넷에 연결-----
retval = bind(serv_sock, (SOCKADDR*)&serv_addr, sizeof(SOCKADDR));
if(retval == SOCKET_ERROR)
{
    printf("bind() Error\n");
    return 0;
}
//-----

//-----대기인원 설정-----
listen(serv_sock, 5); // 5명까지만 대기할 수 있게 함...
//-----
SOCKADDR_IN clnt_addr = {0};
int size = sizeof(SOCKADDR_IN);
while(1)
{
    time(&current_time);
    date=ctime(&current_time);
    //-----클라이언트 접속 대기, connect를 하면 리턴함-----
    SOCKET clnt_sock = accept(serv_sock, (SOCKADDR*)&clnt_addr, &size);
}

```

- accept 함수를 통해 connect 되면 클라이언트와 서버의 접속이 이루어 진다.

3.3 hight와 RSA알고리즘 - 암호화 기능

hight

```

    int number;
    printf(" 파일의 이름을 입력 : ");
    scanf("%s", &name);
NAME:
    printf("1.새로운 비밀번호 입력 #n2.할당된 비밀번호 입력 #n입력 : ");
    scanf("%d", &number);

    FILE* fp = fopen(name, "rb");

    if (number == 2) goto PASS;

    PASS(pbszUserKey); //새로운 패스워드 입력 난수
    printf("#nPASS : %s", pbszUserKey); //입력된 패스워드

    goto TWO; //
PASS: //할당된 비밀번호 입력
    printf("패스워드 입력 :");
    scanf("%s", pbszUserKey);

TWO:
    fpos = ftell(fp);
    fseek(fp, 0, SEEK_END);
    SIZ = ftell(fp);
    fseek(fp, fpos, SEEK_SET);

    printf("SIZ = %d#n#n", SIZ);

    fclose(fp);

    pbszUserKey[16] = NULL;

    printf("1.암,복호화 #n2.RETURN#n3.EXIT#n입력 : ");
    scanf("%d", &number);

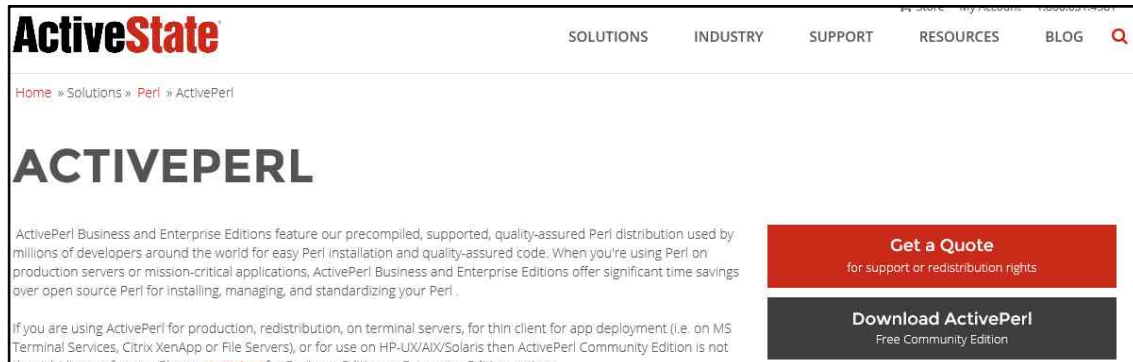
    if (number == 1) goto CYPER;
    if (number == 2) goto NAME; else goto EXIT;

CYPER: //암호화
    CYPER(pbszUserKey, SIZ, name);

```

HIGHT의 일반화된 Feistel 변형 구조로 이루어져 있으며, 64비트의 평문과 128비트 키로부터 생성된 8개의 8비트 화이트닝 키와 128개의 8비트 서브키를 입력으로 사용하여 초기변환과 32회의 라운드변환 그리고 최종변환을 거쳐 64비트 암호문을 출력한다. HIGHT 암호 알고리즘의 장점으로 첫 번째, 기존 SEED, AES 등 알고리즘보다 간단한 알고리즘 구조로 설계되었다. 두 번째, 안정성과 효율성을 동시에 고려한 정교한 암호 알고리즘 개발되었음. 세 번째, HIGHT의 데이터 처리량은 AES보다 15배 이상, 속도는 3배이상이다. 단점으로는 블록수가 작아서 다른 암호화 기법에 비해 보안성이 떨어진다. 하지만 파일을 빠른 속도로 암호화 하는데 있어 많은 작업량이 주어지게 되면 다른 암호화 기법보다 더 빠르게 처리가 가능하다.

RSA 암호화 사용을 위한 OpenSSL 설치과정



www.activestae.com 에서 Active Perl을 다운받고 설치한다.

흔히 cgi라고 불리며 c언어에서 발전된 서버 페이지이며 openssl 바이너리 이전에 필요한 프로그램이다.

```
H:\wopenssl-0.9.8o>perl Configure UC-WIN32 --openssldir=C:\WOpenSSL threads no-asm
Configuring for UC-WIN32
no-asm           [option]      OPENSSL_NO_ASM
no-camellia      [default]    OPENSSL_NO_CAMELLIA <skip dir>
no-capieng       [default]    OPENSSL_NO_CAPIENG <skip dir>
no-cms          [default]    OPENSSL_NO_CMS <skip dir>
no-gmp           [default]    OPENSSL_NO_GMP <skip dir>
no-jpake         [experimental] OPENSSL_NO_JPAKE <skip dir>
no-krb5          [krb5-flavor not specified] OPENSSL_NO_KRB5
no-mdc2          [default]    OPENSSL_NO_MDC2 <skip dir>
no-montasm       [default]
no-rc5           [default]    OPENSSL_NO_RC5 <skip dir>
no-rfc3779       [default]    OPENSSL_NO_RFC3779 <skip dir>
no-seed          [default]    OPENSSL_NO_SEED <skip dir>
no-shared        [default]
no-zlib          [default]
no-zlib-dynamic  [default]

IsMK1MF=1
CC               =cl
CFLAG            =-DOPENSSL_THREADS -DDSO_WIN32
EX_LIBS          =
CPUID_OBJ        =
BN_ASM           =bn_asm.o
DES_ENC          =des_enc.o fcrypt_b.o
AES_ASM_OBJ      =aes_core.o aes_chc.o
BF_ENC           =bf_enc.o
CAST_ENC         =c_enc.o
RC4_ENC          =rc4_enc.o rc4_skey.o
RC5_ENC          =rc5_enc.o
MD5_OBJ_ASM     =
SHA1_OBJ_ASM     =
RMD160_OBJ_ASM  =
PROCESSOR        =
RANLIB           =true
ARFLAGS          =
PERL             =perl
THIRTY_TWO_BIT  mode
BN_LONG mode
RC4_INDEX mode
RC4_CHUNK is undefined

Configured for UC-WIN32.

H:\wopenssl-0.9.8o>
```

관리자 cmd에서 위와 같은 명령어를 입력하면 전반적인 컴파일 과정이 완료된다.

```

H:\Wopenssl-0.9.8k>mswdo_ms.bat

H:\Wopenssl-0.9.8k>perl util\mkfiles.pl 1>MINFO

H:\Wopenssl-0.9.8k>perl util\mk1mf.pl no-asm UC-WIN32 1>msWnt.mak

H:\Wopenssl-0.9.8k>perl util\mk1mf.pl dll no-asm UC-WIN32 1>msWntdll.mak

H:\Wopenssl-0.9.8k>perl util\mk1mf.pl no-asm UC-CE 1>msWce.mak
%OSVERSION% is not defined at util/pl/UC-32.pl line 57.
Compilation failed in require at util\mk1mf.pl line 151.

H:\Wopenssl-0.9.8k>perl util\mk1mf.pl dll no-asm UC-CE 1>msWcedll.mak
%OSVERSION% is not defined at util/pl/UC-32.pl line 57.
Compilation failed in require at util\mk1mf.pl line 151.

H:\Wopenssl-0.9.8k>perl util\mkdef.pl 32 libeay 1>msWlibeay32.def

H:\Wopenssl-0.9.8k>perl util\mkdef.pl 32 ssleay 1>msWssleay32.def

H:\Wopenssl-0.9.8k>

```

또한 구체적인 컴파일 과정으로서 어셈블리를 사용하지 않을 경우 위와 같이 mswdo_ms.bat을 실행시키면 된다.

다음으로 아래와 같이 nmake 명령어로 OpenSSL을 컴파일 한다.

```

H:\Wopenssl-0.9.8k>nmake -f msWntdll.mak install_

```

상위 과정을 거친 OpenSSL 라이브러리를 활용하여 다음과 같이 사용하였다.

```

C:\WINDOWS\system32\cmd.exe
input private_key password : qwer1234
Generating RSA private key, 512 bit long modulus
+++++
.....+++++
e is 65537 (0x10001)
private key create clear
writing RSA key
public key create clear
key.txt file is encrypt by cipher.bin file
key.txt file delete clear

```

```
char cmd[256], private_key[100];

cout << "Input private key password : ";
cin >> private_key;

printf(cmd, "C:\\Users\\李望\\Documents\\李望\\李望\\JB\\李望\\Graduation\\李望\\李望\\bin\\李望\\genrsa -passout pass:Zs -out d:\\李望\\private.pem -des 512", private_key);
system(cmd);

cout << "private key create clear\n";

printf(cmd, "C:\\Users\\李望\\Documents\\李望\\李望\\JB\\李望\\Graduation\\李望\\李望\\bin\\李望\\rsa -passin pass:Zs -in d:\\李望\\private.pem -pubout -out d:\\李望\\public.pem", private_key);
system(cmd);

cout << "public key create clear\n";

printf(cmd, "C:\\Users\\李望\\Documents\\李望\\李望\\JB\\李望\\Graduation\\李望\\李望\\bin\\李望\\rsoutl -encrypt -pubin -inkey d:\\李望\\public.pem -in d:\\李望\\key.txt -out d:\\李望\\cipher.bin");
system(cmd);

cout << "key.txt file is encrypt by cipher.bin file\n";

char strPath[] = ("D:\\李望\\key.txt");

int nResult = remove(strPath);

if (nResult == 0) printf("key.txt file delete clear");
else if (nResult == -1) perror("key.txt file delete clear");

cin >> cmd;

printf(cmd, "C:\\Users\\李望\\Documents\\李望\\李望\\JB\\李望\\Graduation\\李望\\李望\\bin\\李望\\rsoutl -decrypt -inkey d:\\李望\\private.pem -in d:\\李望\\cipher.bin -passin pass:Zs -out d:\\李望\\plain.txt");
system(cmd);
```

openssl 라이브러리를 사용하여 암호화가 가능하게끔 위와 같이 cmd에서 해당툴을 가져와 사용하는 구조를 지님. RSA의 장점은 매우 큰 정수의 소인수분해가 어렵다는 점에서 다른 알고리즘에 비해 안정성이 높고 개인키와 공개키로 비대칭키 형식을 가지므로 암호화에 필요한 공개키가 유출되더라도 파일의 내용을 읽어볼 수 없다. 반대로 단점으로는 안정성이 높은 대신에 매우 큰 정수로 소인수분해를 하기 때문에 암복호화에 걸리는 시간이 오래걸린다. 즉 파일의 크기가 크면 클수록 암복호화 시간이 기하학적으로 늘어난다. 그렇기 때문에 RSA 암호 알고리즘은 개인키를 암호화 하는데 이 기술을 사용하였음.

3.4 주요 기능 코드

- 클라이언트(내 컴퓨터에서 암호화 후 서버로 전송)

case WM_NOTIFY:

```
{
switch(((LPNMHDR)IParam)->code)
{
case NM_DBLCLK:
if (((LPNMHDR)IParam)->idFrom == IDC_LIST)
{
char Text[255]={0};
char Temp[255]={0};
char Temp1[255]={0};
char Text2[255]={0};
char Temp2[255]={0};
char Temp3[255]={0};
char Temp4[255]={"파 일};
char Temp5[255]={"숨김파일};
int iSlected=0;
```

```

    int j=0;

    iSlected=SendMessage(hList,LVM_GETNEXTITEM,-1,LVNI_FOCUSED);
    if(iSlected!=-1)
    {
        MessageBox(hWnd,"No Items in ListView","Error",MB_OK|MB_ICONINFORMATION);
        break;
    }
    memset(&LvItem,0,sizeof(LvItem));
    LvItem.mask=LVIF_TEXT;
    LvItem.iSubItem=1;
    LvItem.pszText=Text;
    LvItem.cchTextMax=256;
    LvItem.iItem=iSlected;

    SendMessage(hList,LVM_GETITEMTEXT, iSlected, (LPARAM)&LvItem);
    sprintf(Temp,"%s", Text);
    lstrcat(Temp1, Temp);
    if (strcmp(Temp1, Temp4) == 0 || strcmp(Temp1, Temp5) == 0)
    {
        if(MessageBox(hWnd, "파일을암호화하시겠습니까, "암호화", MB_YESNO |
        MB_ICONQUESTION) == IDYES)
        {
            memset(&LvItem,0,sizeof(LvItem));
            LvItem.mask=LVIF_TEXT;
            LvItem.iSubItem=0;
            LvItem.pszText=Text;
            LvItem.cchTextMax=256;
            LvItem.iItem=iSlected;

            SendMessage(hList,LVM_GETITEMTEXT, iSlected, (LPARAM)&LvItem);
            sprintf(Temp2,"%s", Text);
            lstrcat(Temp3, Temp2);
            char *backup[100], c_size[100];
            int level = 0, i;
            int size;

```

```

    path[level] = (char *)calloc(PSIZE, sizeof(char));
    strcpy_s(path[level], 256, "c:\\test");
    strcat_s(path[level], 256, "\\");
    strcat_s(path[level], 256, Temp3);
    backup[level] = (char *)calloc(PSIZE, sizeof(char));
    strcpy_s(backup[level], 1024, path[level]);

    PASS(pbszUserKey);
    CYPER(pbszUserKey, backup, Temp3);
    MessageBox(hWnd, "파일암호화완료, "파일암호화, MB_ICONASTERISK);

    WSADATA wsaData;
    int retval = WSStartup(MAKEWORD(2,2), &wsaData);
    if(retval != 0)
    {
        printf("WSAStartup() Error\n");
        return 0;
    }

    SOCKET serv_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    SOCKADDR_IN serv_addr;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(4000);
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    retval = connect(serv_sock, (SOCKADDR*)&serv_addr, sizeof(SOCKADDR));
    if(retval == SOCKET_ERROR)
    {
        printf("connect() Error\n");
        return 0;
    }

    char t_filename[256], *plaintext, c;
    strcpy(t_filename, "C:\\Users\\정세욱(2) (2)\\");
    strcat(t_filename, Temp3);
    ifstream fsrc(Temp3, ios::in | ios::binary);
    FILE* fin = fopen(Temp3, "r");

```

```

fseek( fin, 0, SEEK_END );
size = ftell( fin );
_itoa(size, c_size, 10);
plaintext= (char *)calloc(size+1, sizeof(char));

for(i=0; i<size; i++)
{
    c= fsrc.get();
    plaintext[i]=c;
}

char buf[256];
char buf2[1024]={0};
strcpy(buf, "4:");
strcat(buf, str);
strcat(buf, ":");
strcat(buf, Temp3);
strcat(buf, ":");
strcat(buf, c_size);
strcat(buf, ":");

int sendsize = send(serv_sock,buf,strlen(buf),0);

if(sendsize <= 0)break;
sendsize = send(serv_sock, plaintext, size, 0);
fsrc.close();
closesocket(serv_sock);
WSACleanup();
strtok(path[level], Temp3);
}

```



〈파일 암호화 및 송신 과정〉

- 클라이언트(서버에 저장된 파일 복호화 방법)

```
else if (((LPNMHDR)IParam)->idFrom == IDC_LIST2)
```

```
{
```

```
char Text[255]={0};
```

```
char Temp[255]={0};
```

```
char Temp1[255]={0};
```

```
char Text2[255]={0};
```

```
char Temp2[255]={0};
```

```
char Temp3[255]={0};
```

```
char T_size[255]={0};
```

```
char T_size1[255]={0};
```

```
char Temp4[255]={"파일"};
```

```
char Temp5[255]={"숨김파일"};
```

```
int iSlected2=0;
```

```
int j=0;
```

```
iSlected2=SendMessage(hList2,LVM_GETNEXTITEM,-1,LVNI_FOCUSED);
```

```
if(iSlected2==-1)
```

```
{
```

```
MessageBox(hWnd,"No Items in ListView","Error",MB_OK|MB_ICONINFORMATION);
```

```
break;
```

```
}
```

```
memset(&LvItem2,0,sizeof(LvItem2));
```

```
LvItem2.mask=LVIF_TEXT;
```

```
LvlItem2.iSubItem=1;
LvlItem2.pszText=Text;
LvlItem2.cchTextMax=256;
LvlItem2.iItem=iSlected2;
```

```
SendMessage(hList2,LVM_GETITEMTEXT, iSlected2, (LPARAM)&LvlItem2);
sprintf(Temp,"%s", Text);
lstrcat(Temp1, Temp);
if (strcmp(Temp1, Temp4) == 0 || strcmp(Temp1, Temp5) == 0)
{
if(MessageBox(hWnd, "파일을 복호화 하시겠습니까, "복호화", MB_YESNO |
MB_ICONQUESTION) == IDYES)
{
memset(&LvlItem,0,sizeof(LvlItem));
LvlItem.mask=LVIF_TEXT;
LvlItem.iSubItem=0;
LvlItem.pszText=Text;
LvlItem.cchTextMax=256;
LvlItem.iItem=iSlected2;
```

```
SendMessage(hList2,LVM_GETITEMTEXT, iSlected2, (LPARAM)&LvlItem);
sprintf(Temp2,"%s", Text);
lstrcat(Temp3, Temp2);
memset(&LvlItem,0,sizeof(LvlItem));
LvlItem.mask=LVIF_TEXT;
LvlItem.iSubItem=2;
LvlItem.pszText=Text;
LvlItem.cchTextMax=256;
LvlItem.iItem=iSlected2;
```

```
SendMessage(hList2,LVM_GETITEMTEXT, iSlected2, (LPARAM)&LvlItem);
sprintf(T_size1,"%s", Text);
lstrcat(T_size, T_size1);
MessageBox(hWnd, T_size, T_size, MB_OK);
```

```
WSADATA wsaData;
int retval = WSStartup(MAKEWORD(2,2),&wsaData);
```

```

    if(retval != 0)
    {
        printf("WSAStartup() Error\n");
        return 0;
    }
    SOCKET serv_sock = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    SOCKADDR_IN serv_addr;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(4000);
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    retval = connect(serv_sock,(SOCKADDR*)&serv_addr,sizeof(SOCKADDR));
    if(retval == SOCKET_ERROR)
    {
        printf("connect() Error\n");
        return 0;
    }

    char buf[256];
    char buf2[1024]={0};

    strcpy(buf, "5:");
    strcat(buf, str);
    strcat(buf, ":");
    strcat(buf, Temp3);
    strcat(buf, ":");
    char r_filename[256];
    int size, i, recvsize;

    strcpy(r_filename,"C:\\test\\");
    strcat(r_filename,Temp3);
    size = atoi(T_size);
    char *buffile;
    buffile = (char *)calloc(size, sizeof(char));
    MessageBox(hWnd, r_filename, r_filename, MB_OK);
    int sendsize = send(serv_sock,buf,strlen(buf),0);
    if(sendsize <= 0)break;

```

```

while(1){
recvsize = recvn(serv_sock, buffile, size, 0);
if (recvsize == 0)break;
}
MessageBox(hWnd, r_filename, r_filename, MB_OK);
setlocale(LC_ALL, "korean");
ofstream fdest(r_filename, ios::out | ios::binary);
MessageBox(hWnd, T_size, T_size, MB_OK);
for(i = 0; i < size; i++)
{
    fdest << buffile[i];
}
fdest.close();
free(buffile);
closesocket(serv_sock);
WSACleanup();

```



〈파일 복호화 과정〉

- 서버(서버에 저장된 유저들 받아오기)

```

hList3 = GetDlgItem(hWnd, IDC_COMBO1);
for(j=0; j<i; j++)
{
    SendMessage(hList3, CB_ADDSTRING, 0, (LPARAM)save3[j]);
}
SendMessage(hList3, CB_SETCURSEL, 0, 0);

```



〈콤보 박스를 통한 유저 리스트 출력〉

- 파일 암호화 방식

```
#include "KISA_HIGHT_CTR.h"
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
#include <iostream>
```

```
unsigned char F0[256] =
```

```
{
```

```
    0x00, 0x86, 0x0D, 0x8B, 0x1A, 0x9C, 0x17, 0x91,  
    0x34, 0xB2, 0x39, 0xBF, 0x2E, 0xA8, 0x23, 0xA5,  
    0x68, 0xEE, 0x65, 0xE3, 0x72, 0xF4, 0x7F, 0xF9,  
    0x5C, 0xDA, 0x51, 0xD7, 0x46, 0xC0, 0x4B, 0xCD,  
    0xD0, 0x56, 0xDD, 0x5B, 0xCA, 0x4C, 0xC7, 0x41,  
    0xE4, 0x62, 0xE9, 0x6F, 0xFE, 0x78, 0xF3, 0x75,  
    0xB8, 0x3E, 0xB5, 0x33, 0xA2, 0x24, 0xAF, 0x29,  
    0x8C, 0x0A, 0x81, 0x07, 0x96, 0x10, 0x9B, 0x1D,  
    0xA1, 0x27, 0xAC, 0x2A, 0xBB, 0x3D, 0xB6, 0x30,  
    0x95, 0x13, 0x98, 0x1E, 0x8F, 0x09, 0x82, 0x04,  
    0xC9, 0x4F, 0xC4, 0x42, 0xD3, 0x55, 0xDE, 0x58,  
    0xFD, 0x7B, 0xF0, 0x76, 0xE7, 0x61, 0xEA, 0x6C,  
    0x71, 0xF7, 0x7C, 0xFA, 0x6B, 0xED, 0x66, 0xE0,  
    0x45, 0xC3, 0x48, 0xCE, 0x5F, 0xD9, 0x52, 0xD4,  
    0x19, 0x9F, 0x14, 0x92, 0x03, 0x85, 0x0E, 0x88,  
    0x2D, 0xAB, 0x20, 0xA6, 0x37, 0xB1, 0x3A, 0xBC,  
    0x43, 0xC5, 0x4E, 0xC8, 0x59, 0xDF, 0x54, 0xD2,  
    0x77, 0xF1, 0x7A, 0xFC, 0x6D, 0xEB, 0x60, 0xE6,  
    0x2B, 0xAD, 0x26, 0xA0, 0x31, 0xB7, 0x3C, 0xBA,  
    0x1F, 0x99, 0x12, 0x94, 0x05, 0x83, 0x08, 0x8E,  
    0x93, 0x15, 0x9E, 0x18, 0x89, 0x0F, 0x84, 0x02,  
    0xA7, 0x21, 0xAA, 0x2C, 0xBD, 0x3B, 0xB0, 0x36,  
    0xFB, 0x7D, 0xF6, 0x70, 0xE1, 0x67, 0xEC, 0x6A,  
    0xCF, 0x49, 0xC2, 0x44, 0xD5, 0x53, 0xD8, 0x5E,  
    0xE2, 0x64, 0xEF, 0x69, 0xF8, 0x7E, 0xF5, 0x73,  
    0xD6, 0x50, 0xDB, 0x5D, 0xCC, 0x4A, 0xC1, 0x47,  
    0x8A, 0x0C, 0x87, 0x01, 0x90, 0x16, 0x9D, 0x1B,  
    0xBE, 0x38, 0xB3, 0x35, 0xA4, 0x22, 0xA9, 0x2F,
```

```
    0x32, 0xB4, 0x3F, 0xB9, 0x28, 0xAE, 0x25, 0xA3,  
    0x06, 0x80, 0x0B, 0x8D, 0x1C, 0x9A, 0x11, 0x97,  
    0x5A, 0xDC, 0x57, 0xD1, 0x40, 0xC6, 0x4D, 0xCB,  
    0x6E, 0xE8, 0x63, 0xE5, 0x74, 0xF2, 0x79, 0xFF  
};
```

```
unsigned char F1[256] =
```

```
{  
    0x00, 0x58, 0xB0, 0xE8, 0x61, 0x39, 0xD1, 0x89,  
    0xC2, 0x9A, 0x72, 0x2A, 0xA3, 0xFB, 0x13, 0x4B,  
    0x85, 0xDD, 0x35, 0x6D, 0xE4, 0xBC, 0x54, 0x0C,  
    0x47, 0x1F, 0xF7, 0xAF, 0x26, 0x7E, 0x96, 0xCE,  
    0x0B, 0x53, 0xBB, 0xE3, 0x6A, 0x32, 0xDA, 0x82,  
    0xC9, 0x91, 0x79, 0x21, 0xA8, 0xF0, 0x18, 0x40,  
    0x8E, 0xD6, 0x3E, 0x66, 0xEF, 0xB7, 0x5F, 0x07,  
    0x4C, 0x14, 0xFC, 0xA4, 0x2D, 0x75, 0x9D, 0xC5,  
    0x16, 0x4E, 0xA6, 0xFE, 0x77, 0x2F, 0xC7, 0x9F,  
    0xD4, 0x8C, 0x64, 0x3C, 0xB5, 0xED, 0x05, 0x5D,  
    0x93, 0xCB, 0x23, 0x7B, 0xF2, 0xAA, 0x42, 0x1A,  
    0x51, 0x09, 0xE1, 0xB9, 0x30, 0x68, 0x80, 0xD8,  
    0x1D, 0x45, 0xAD, 0xF5, 0x7C, 0x24, 0xCC, 0x94,  
    0xDF, 0x87, 0x6F, 0x37, 0xBE, 0xE6, 0x0E, 0x56,  
    0x98, 0xC0, 0x28, 0x70, 0xF9, 0xA1, 0x49, 0x11,  
    0x5A, 0x02, 0xEA, 0xB2, 0x3B, 0x63, 0x8B, 0xD3,  
    0x2C, 0x74, 0x9C, 0xC4, 0x4D, 0x15, 0xFD, 0xA5,  
    0xEE, 0xB6, 0x5E, 0x06, 0x8F, 0xD7, 0x3F, 0x67,  
    0xA9, 0xF1, 0x19, 0x41, 0xC8, 0x90, 0x78, 0x20,  
    0x6B, 0x33, 0xDB, 0x83, 0x0A, 0x52, 0xBA, 0xE2,  
    0x27, 0x7F, 0x97, 0xCF, 0x46, 0x1E, 0xF6, 0xAE,  
    0xE5, 0xBD, 0x55, 0x0D, 0x84, 0xDC, 0x34, 0x6C,  
    0xA2, 0xFA, 0x12, 0x4A, 0xC3, 0x9B, 0x73, 0x2B,  
    0x60, 0x38, 0xD0, 0x88, 0x01, 0x59, 0xB1, 0xE9,  
    0x3A, 0x62, 0x8A, 0xD2, 0x5B, 0x03, 0xEB, 0xB3,  
    0xF8, 0xA0, 0x48, 0x10, 0x99, 0xC1, 0x29, 0x71,  
    0xBF, 0xE7, 0x0F, 0x57, 0xDE, 0x86, 0x6E, 0x36,  
    0x7D, 0x25, 0xCD, 0x95, 0x1C, 0x44, 0xAC, 0xF4,  
    0x31, 0x69, 0x81, 0xD9, 0x50, 0x08, 0xE0, 0xB8,  
}
```

```

    0xF3, 0xAB, 0x43, 0x1B, 0x92, 0xCA, 0x22, 0x7A,
    0xB4, 0xEC, 0x04, 0x5C, 0xD5, 0x8D, 0x65, 0x3D,
    0x76, 0x2E, 0xC6, 0x9E, 0x17, 0x4F, 0xA7, 0xFF
};

#define BLOCK_SIZE_HIGHT      8
#define BLOCK_SIZE_HIGHT_INT 2

unsigned char Delta[128] =
{
    0x5a, 0x6d, 0x36, 0x1b, 0x0d, 0x06, 0x03, 0x41, 0x60, 0x30,
    0x18, 0x4c, 0x66, 0x33, 0x59, 0x2c,
    0x56, 0x2b, 0x15, 0x4a, 0x65, 0x72, 0x39, 0x1c, 0x4e, 0x67,
    0x73, 0x79, 0x3c, 0x5e, 0x6f, 0x37,
    0x5b, 0x2d, 0x16, 0x0b, 0x05, 0x42, 0x21, 0x50, 0x28, 0x54,
    0x2a, 0x55, 0x6a, 0x75, 0x7a, 0x7d,
    0x3e, 0x5f, 0x2f, 0x17, 0x4b, 0x25, 0x52, 0x29, 0x14, 0x0a,
    0x45, 0x62, 0x31, 0x58, 0x6c, 0x76,
    0x3b, 0x1d, 0x0e, 0x47, 0x63, 0x71, 0x78, 0x7c, 0x7e, 0x7f,
    0x3f, 0x1f, 0x0f, 0x07, 0x43, 0x61,
    0x70, 0x38, 0x5c, 0x6e, 0x77, 0x7b, 0x3d, 0x1e, 0x4f, 0x27,
    0x53, 0x69, 0x34, 0x1a, 0x4d, 0x26,
    0x13, 0x49, 0x24, 0x12, 0x09, 0x04, 0x02, 0x01, 0x40, 0x20,
    0x10, 0x08, 0x44, 0x22, 0x11, 0x48,
    0x64, 0x32, 0x19, 0x0c, 0x46, 0x23, 0x51, 0x68, 0x74, 0x3a,
    0x5d, 0x2e, 0x57, 0x6b, 0x35, 0x5a
};

#define BLOCK_XOR_HIGHT( OUT_VALUE, IN_VALUE1, IN_VALUE2 ) {
    \
    OUT_VALUE[0] = IN_VALUE1[0] ^ IN_VALUE2[0];
    \
    OUT_VALUE[1] = IN_VALUE1[1] ^ IN_VALUE2[1];
    \
}

void UpdateCounter_for_HIGHT(unsigned char *pbOUT, int nIncreaseValue, int nMin)

```

```

{
    unsigned char bszBackup = 0;
    int i;

    if (0 > nMin)
        return

    if (0 < nMin)
    {
        bszBackup = pbOUT[nMin];
        pbOUT[nMin] += nIncreaseValue;
    }

    for (i = nMin; i>1; --i)
    {
        if (bszBackup <= pbOUT[i])
            return
        else
        {
            bszBackup = pbOUT[i - 1];
            pbOUT[i - 1] += 1;
        }
    }
    bszBackup = pbOUT[0];
    pbOUT[0] += nIncreaseValue;
}

#define EncIni_Transformation(x0,x2,x4,x6,mk0,mk1,mk2,mk3) \
    t0 = x0 + mk0; \
    t2 = x2 ^ mk1; \
    t4 = x4 + mk2; \
    t6 = x6 ^ mk3;

#define EncFin_Transformation(x0,x2,x4,x6,mk0,mk1,mk2,mk3) \

```

```

out[0] = x0 + mk0;
      \
out[2] = x2 ^ mk1;
      \
out[4] = x4 + mk2;
      \
out[6] = x6 ^ mk3;

```

```

#define Round(x7,x6,x5,x4,x3,x2,x1,x0)

```

```

\
x1 += (F1[x0] ^ key[0]);
      \
x3 ^= (F0[x2] + key[1]);
      \
x5 += (F1[x4] ^ key[2]);
      \
x7 ^= (F0[x6] + key[3]);

```

```

void KISA_HIGHT_Block_forCTR(unsigned char *pbszIN_Key128,
unsigned char *pbszUserKey, const unsigned char *in, unsigned
char *out)

```

```

{

```

```

    register unsigned char t0, t1, t2, t3, t4, t5, t6, t7;
    unsigned char *key, *key2;

```

```

    key = pbszIN_Key128;
    key2 = pbszUserKey;

```

```

    t1 = in[1]; t3 = in[3]; t5 = in[5]; t7 = in[7];
    Enclni_Transformation(in[0], in[2], in[4], in[6], key2[12], key2[13],
                           key2[14], key2[15]);

```

```

    Round(t7, t6, t5, t4, t3, t2, t1, t0); key += 4;           // 1
    Round(t6, t5, t4, t3, t2, t1, t0, t7); key += 4;          // 2
    Round(t5, t4, t3, t2, t1, t0, t7, t6); key += 4;          // 3
    Round(t4, t3, t2, t1, t0, t7, t6, t5); key += 4;          // 4
    Round(t3, t2, t1, t0, t7, t6, t5, t4); key += 4;          // 5

```

```

Round(t2, t1, t0, t7, t6, t5, t4, t3); key += 4; // 6
Round(t1, t0, t7, t6, t5, t4, t3, t2); key += 4; // 7
Round(t0, t7, t6, t5, t4, t3, t2, t1); key += 4; // 8
Round(t7, t6, t5, t4, t3, t2, t1, t0); key += 4; // 9
Round(t6, t5, t4, t3, t2, t1, t0, t7); key += 4; // 10
Round(t5, t4, t3, t2, t1, t0, t7, t6); key += 4; // 11
Round(t4, t3, t2, t1, t0, t7, t6, t5); key += 4; // 12
Round(t3, t2, t1, t0, t7, t6, t5, t4); key += 4; // 13
Round(t2, t1, t0, t7, t6, t5, t4, t3); key += 4; // 14
Round(t1, t0, t7, t6, t5, t4, t3, t2); key += 4; // 15
Round(t0, t7, t6, t5, t4, t3, t2, t1); key += 4; // 16
Round(t7, t6, t5, t4, t3, t2, t1, t0); key += 4; // 17
Round(t6, t5, t4, t3, t2, t1, t0, t7); key += 4; // 18
Round(t5, t4, t3, t2, t1, t0, t7, t6); key += 4; // 19
Round(t4, t3, t2, t1, t0, t7, t6, t5); key += 4; // 20
Round(t3, t2, t1, t0, t7, t6, t5, t4); key += 4; // 21
Round(t2, t1, t0, t7, t6, t5, t4, t3); key += 4; // 22
Round(t1, t0, t7, t6, t5, t4, t3, t2); key += 4; // 23
Round(t0, t7, t6, t5, t4, t3, t2, t1); key += 4; // 24
Round(t7, t6, t5, t4, t3, t2, t1, t0); key += 4; // 25
Round(t6, t5, t4, t3, t2, t1, t0, t7); key += 4; // 26
Round(t5, t4, t3, t2, t1, t0, t7, t6); key += 4; // 27
Round(t4, t3, t2, t1, t0, t7, t6, t5); key += 4; // 28
Round(t3, t2, t1, t0, t7, t6, t5, t4); key += 4; // 29
Round(t2, t1, t0, t7, t6, t5, t4, t3); key += 4; // 30
Round(t1, t0, t7, t6, t5, t4, t3, t2); key += 4; // 31
Round(t0, t7, t6, t5, t4, t3, t2, t1); // 32

```

```

EncFin_Transformation(t1, t3, t5, t7, key2[0], key2[1], key2[2], key2[3]);
out[1] = t2; out[3] = t4; out[5] = t6; out[7] = t0;

```

```

}

```

```

void InitNonce_HIGHT(IN NONCE_TYPE type, IN unsigned char
*pbszIV, IN unsigned char *pbszCounter, OUT unsigned char
*pbszNonce)
{
    switch (type)

```

```
{  
  case NONCE_OR:  
  {  
    pbszNonce[0] = pbszIV[0] | pbszCounter[0];  
    pbszNonce[1] = pbszIV[1] | pbszCounter[1];  
    pbszNonce[2] = pbszIV[2] | pbszCounter[2];  
    pbszNonce[3] = pbszIV[3] | pbszCounter[3];  
    pbszNonce[4] = pbszIV[4] | pbszCounter[4];  
    pbszNonce[5] = pbszIV[5] | pbszCounter[5];  
    pbszNonce[6] = pbszIV[6] | pbszCounter[6];  
    pbszNonce[7] = pbszIV[7] | pbszCounter[7];  
  }  
  break;  
  case NONCE_AND:  
  {  
    pbszNonce[0] = pbszIV[0] & pbszCounter[0];  
    pbszNonce[1] = pbszIV[1] & pbszCounter[1];  
    pbszNonce[2] = pbszIV[2] & pbszCounter[2];  
    pbszNonce[3] = pbszIV[3] & pbszCounter[3];  
    pbszNonce[4] = pbszIV[4] & pbszCounter[4];  
    pbszNonce[5] = pbszIV[5] & pbszCounter[5];  
    pbszNonce[6] = pbszIV[6] & pbszCounter[6];  
    pbszNonce[7] = pbszIV[7] & pbszCounter[7];  
  }  
  break;  
  case NONCE_XOR:  
  {  
    pbszNonce[0] = pbszIV[0] ^ pbszCounter[0];  
    pbszNonce[1] = pbszIV[1] ^ pbszCounter[1];  
    pbszNonce[2] = pbszIV[2] ^ pbszCounter[2];  
    pbszNonce[3] = pbszIV[3] ^ pbszCounter[3];  
    pbszNonce[4] = pbszIV[4] ^ pbszCounter[4];  
    pbszNonce[5] = pbszIV[5] ^ pbszCounter[5];  
    pbszNonce[6] = pbszIV[6] ^ pbszCounter[6];  
    pbszNonce[7] = pbszIV[7] ^ pbszCounter[7];  
  }  
  break;
```

```
    }  
}
```

```
unsigned int * chartoint32_for_HIGHT_CTR(IN unsigned char *in, IN int inLen)  
{  
    unsigned int *data;  
    int len, i;  
  
    if (inLen % 4)  
        len = (inLen / 4) + 1;  
    else  
        len = (inLen / 4);  
  
    data =(unsigned int *)malloc(sizeof(unsigned int)* len);  
  
    for (i = 0; i<len; i++)  
    {  
        data[i] = ((unsigned int*)in)[i];  
    }  
  
    return data;  
}
```

```
unsigned char* int32tochar_for_HIGHT_CTR(IN unsigned int *in, IN int inLen)  
{  
    unsigned char *data;  
    int i;  
  
    data = (unsigned char *)malloc(sizeof(unsigned char)* inLen);  
  
#ifndef BIG_ENDIAN  
    for (i = 0; i<inLen; i++)  
    {  
        data[i] = (unsigned char)(in[i / 4] >> ((i % 4) * 8));  
    }  
#else  
    for (i = 0; i<inLen; i++)
```

```

        {
            data[i] = (unsigned char)(in[i / 4] >> ((3 - (i % 4)) *
8));
        }
    #endif

    return data;
}

void HIGHT_CTR_init(OUT KISA_HIGHT_INFO *pInfo, IN
KISA_ENC_DEC enc, IN unsigned char *pUserKey, IN unsigned char
*pszbIV)
{
    unsigned char i, j;

    memset(pInfo, 0, sizeof(KISA_HIGHT_INFO));
    pInfo->encrypt = enc;
    memcpy((unsigned char *)pInfo->ctr, (unsigned char *)pszbIV,
BLOCK_SIZE_HIGHT);
    memcpy(pInfo->userKey, pUserKey, 16);

    for (i = 0; i < BLOCK_SIZE_HIGHT; i++)
    {
        for (j = 0; j < BLOCK_SIZE_HIGHT; j++)
            pInfo->hight_key.key_data[16 * i + j] = pUserKey[(j - i)
& 7] + Delta[16 * i + j];

        for (j = 0; j < BLOCK_SIZE_HIGHT; j++)
            pInfo->hight_key.key_data[16 * i + j + 8] = pUserKey[((j
- i) & 7) + 8] + Delta[16 * i + j + 8];
    }
}

int HIGHT_CTR_Process(OUT KISA_HIGHT_INFO *pInfo, IN unsigned
int *in, IN int inLen, OUT unsigned int *out, OUT int *outLen)
{
    unsigned char *pbszCounter;

```

```

    unsigned int *pdwCounter;
    unsigned int *pdwFirst;
    int nCurrentCount = 0;

    if (NULL == pInfo ||
        NULL == in ||
        NULL == out ||
        0 > inLen)
        return 0;

    pdwCounter = pInfo->ctr;

    while (nCurrentCount < inLen)
    {
        KISA_HIGHT_Block_forCTR(pInfo->hight_key.key_data,
pInfo->userKey, (unsigned char *)pdwCounter, (unsigned char
*)out);

        BLOCK_XOR_HIGHT(out, in, out);

        if (0 == nCurrentCount)pdwFirst = out;

        pbszCounter = (unsigned char *)pdwCounter;
        UpdateCounter_for_HIGHT(pbszCounter, 1, (BLOCK_SIZE_HIGHT - 1));

        nCurrentCount += BLOCK_SIZE_HIGHT;
        in += BLOCK_SIZE_HIGHT_INT;
        out += BLOCK_SIZE_HIGHT_INT;
    }

    *outLen = nCurrentCount;
    pInfo->buffer_length = inLen - *outLen;

    return 1;
}

int HIGHT_CTR_Close(OUT KISA_HIGHT_INFO *pInfo, IN unsigned int
*out, IN int out_offset, IN int *outLen)

```

```

{
    unsigned int nEncRmainLeng;
    int i;
    unsigned char *pOut;
    out += (out_offset / 4);
    pOut = (unsigned char *)(out);

    nEncRmainLeng = -(pInfo->buffer_length);

    for (i = nEncRmainLeng; i>0; i--)
    {
        *(pOut - i) = (unsigned char)0x00;
    }

    *outLen = nEncRmainLeng;

    return 1;
}

```

```

int HIGHT_CTR_Encrypt(IN unsigned char *pbszUserKey, IN unsigned
char *pszbCounter, IN unsigned char *pbInputText, IN int in_offset,
IN int nInputTextLen, OUT unsigned char *pbszOutputText)
{
    int nOutLeng = 0;
    int nEncRmainLeng = 0;
    KISA_HIGHT_INFO info;
    unsigned char *newpbszInputText;
    unsigned int *outbuf;
    unsigned int *data;
    unsigned char *cdata;
    int outlen = 0;

    int nInputTextPadding = (BLOCK_SIZE_HIGHT - (nInputTextLen
% BLOCK_SIZE_HIGHT)) % BLOCK_SIZE_HIGHT;
    newpbszInputText = (unsigned char *)malloc(sizeof(unsigned
char)* (nInputTextLen + nInputTextPadding));
    memcpy(newpbszInputText, pbInputText + in_offset, nInputTextLen);

```

```

HIGHT_CTR_init(&info, KISA_ENCRYPT, pbszUserKey, pszbCounter);
outlen = ((nInputTextLen + nInputTextPadding) / 8) * 2;
outbuf = (unsigned int *)malloc(sizeof(unsigned int)* outlen);
data = chartoint32_for_HIGHT_CTR(newpbszInputText, nInputTextLen);

HIGHT_CTR_Process(&info, data, nInputTextLen, outbuf, &nOutLeng);
HIGHT_CTR_Close(&info, outbuf, nOutLeng, &nEncRmainLeng);
cdata = int32tochar_for_HIGHT_CTR(outbuf, nOutLeng - nEncRmainLeng);
memcpy(pbszOutputText, cdata, nOutLeng - nEncRmainLeng);

free(data);
free(cdata);
free(outbuf);

return (nOutLeng - nEncRmainLeng);
}

```

```

int HIGHT_CTR_Decrypt(IN unsigned char *pbszUserKey, IN unsigned
char *pszbCounter, IN unsigned char *pbInputText, IN int in_offset,
IN int nInputTextLen, OUT unsigned char *pbszOutputText)
{
    int nOutLeng = 0;
    int nEncRmainLeng = 0;
    KISA_HIGHT_INFO info;
    unsigned char *newpbszInputText;
    unsigned int *outbuf;
    unsigned int *data;
    unsigned char *cdata;
    int outlen = 0;

    int nInputTextPadding = (BLOCK_SIZE_HIGHT - (nInputTextLen
% BLOCK_SIZE_HIGHT)) % BLOCK_SIZE_HIGHT;
    newpbszInputText = (unsigned char *)malloc(sizeof(unsigned
char)* (nInputTextLen + nInputTextPadding));
    memcpy(newpbszInputText, pbInputText + in_offset, nInputTextLen);
    HIGHT_CTR_init(&info, KISA_ENCRYPT, pbszUserKey, pszbCounter);

```

```

    outlen = ((nInputTextLen + nInputTextPadding) / 8) * 2;

    outbuf = (unsigned int *)malloc(sizeof(unsigned int)* outlen);
    data = chartoint32_for_HIGHT_CTR(newpbszInputText, nInputTextLen);

    HIGHT_CTR_Process(&info, data, nInputTextLen, outbuf, &nOutLeng);
    HIGHT_CTR_Close(&info, outbuf, nOutLeng, &nEncRmainLeng);
    cdata = int32tochar_for_HIGHT_CTR(outbuf, nOutLeng - nEncRmainLeng);
    memcpy(pbszOutputText, cdata, nOutLeng - nEncRmainLeng);

    free(data);
    free(cdata);
    free(outbuf);

    return (nOutLeng - nEncRmainLeng);
}

void CYPER(unsigned char pbszUserKey[17] ,char *name[100], char
filename[100]) ////////////
{
    //암호화
    unsigned char pbszCounter[8] = { 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xfe };
    int i;
    int nInputTextLen;
    int message_length = 0;
    int nOutputTextLen = 0;
    int SIZ;
    KISA_HIGHT_INFO info;
    unsigned int *outbuf;
#define process_blockLeng      16
    unsigned int *data;
    unsigned char *cdata;
    int nOutLeng = 0;
    int remainleng = 0;
    int nEncRmainLeng = 0;
    int modmod;

```

```

int fpos;

unsigned char *OutputBytes;
unsigned char *pbszInputText;
unsigned char *InputBytes ;
// unsigned char *OutputBytes;
// //////////////////////////////////////
FILE* fp = fopen(*name, "rb"); //본파일열기
FILE* wp = fopen(filename, "wb"); // 암호화파일생성하며열기

    fpos = ftell(fp);
    fseek(fp, 0, SEEK_END);
    SIZ = ftell(fp);
    fseek(fp, fpos, SEEK_SET); //본파일의크기

    if( 0 != SIZ % 16) modmod = SIZ % 16 ;
    else modmod = 16;
    SIZ = ceil((float)SIZ / 16);

    nInputTextLen = SIZ;

for (int k = 0; k < SIZ; k++)
{
    pbszInputText = (unsigned char *)calloc(16, sizeof(char));
    OutputBytes = (unsigned char *)calloc(32, sizeof(char));
    InputBytes = (unsigned char *)calloc(16, sizeof(char));

    for (int j = 0; j < 16; j++)
    {
        if (feof(fp)) break
        InputBytes[j] = fgetc(fp);
    }

    if (k == SIZ-1) message_length = modmod;
    else message_length = 16;
}

```

```

if (message_length == 0)
{

    nOutputTextLen = 0;
}
else
{

HIGHT_CTR_init(&info, KISA_ENCRYPT, pbszUserKey, pbszCounter);
outbuf = (unsigned int *)malloc(sizeof(unsigned int)* process_blockLeng);

for (i = 0; i < message_length - process_blockLeng;)
{
memcpy(pbszInputText, InputBytes + i, process_blockLeng);
data = chartoint32_for_HIGHT_CTR(pbszInputText, process_blockLeng);
HIGHT_CTR_Process(&info, data, process_blockLeng, outbuf, &nOutLeng);
cdata = int32tochar_for_HIGHT_CTR(outbuf, nOutLeng);
memcpy(OutputBytes + i, cdata, nOutLeng);
    i += nOutLeng;
    free(data);
    free(cdata);
}

remainleng = message_length % process_blockLeng;
if (remainleng == 0)
{
    remainleng = process_blockLeng;
}

memcpy(pbszInputText, InputBytes + i, remainleng);
data = chartoint32_for_HIGHT_CTR(pbszInputText, remainleng);
HIGHT_CTR_Process(&info, data, remainleng, outbuf, &nOutLeng);
HIGHT_CTR_Close(&info, outbuf, nOutLeng, &nEncRmainLeng);
cdata = int32tochar_for_HIGHT_CTR(outbuf, nOutLeng - nEncRmainLeng);
memcpy(OutputBytes + i, cdata, nOutLeng - nEncRmainLeng);
    free(data);
    free(cdata);
}

```

```
OutputBytes[message_length] = NULL;
```

```
for (int j = 0; j < message_length; j++)  
{  
    fputc(OutputBytes[j],wp);  
}
```

```
nOutputTextLen = i + nOutLeng - nEncRmainLeng;
```

```
free(pbszInputText);
```

```
free(InputBytes);
```

```
free(OutputBytes);
```

```
free(outbuf);
```

```
}
```

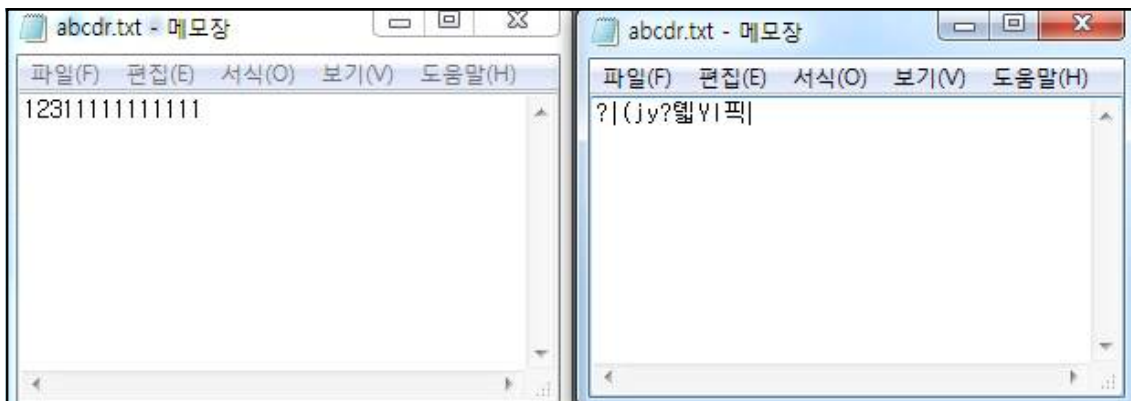
```
}
```

```
fclose(fp);
```

```
fclose(wp);
```

```
}
```

- 메모장, MP3, PPT, 한글문서, 동영상 등 다양한 프로그램 암호화가 가능함.



〈파일 암호화 결과〉

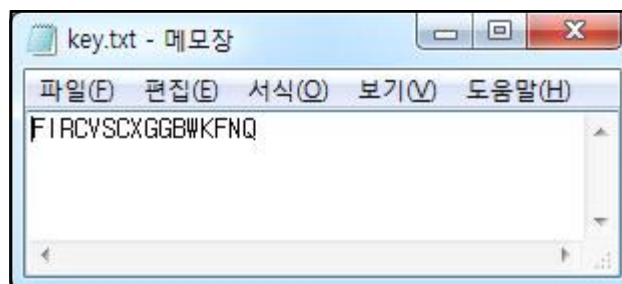
- 개인키 발급

```
#include "KISA_HIGHT_CTR.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "time.h"
#pragma warning(disable:4996)
void PASS(unsigned char *pbszUserKey) //패스워드난수
{
    int i = 0;
    //srand(time(NULL));
    for (i = 0; i < 16; i++)
    {
        pbszUserKey[i] = ((rand() % 26) + 65);

        printf("key %d = %c\n ", i, pbszUserKey[i]);
    }
    pbszUserKey[16] = NULL;

    FILE* PASS = fopen("key.txt", "w");

    for (i = 0; i < 16; i++)
    {
        fputc(pbszUserKey[i], PASS);
        // printf("%c ", OutputBytes[i]);
    }
    pbszUserKey[16] = NULL;
    fclose(PASS);
}
```



<개인키 발급>

- 서버 구현

```
#define _CRT_SECURE_NO_WARNINGS
#pragma comment (lib , "ws2_32.lib")
#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>
#include <process.h>
#include <io.h>
#include <time.h>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <locale>
#include <windows.h>

using namespace std;
#define PSIZE 65536

int recvn(SOCKET s, char *buf, int len, int flags)
{
    int received;
    char *ptr = buf;
    int left = len;

    while (left > 0){
        received = recv(s, ptr, left, flags);
        if (received == SOCKET_ERROR)
            return SOCKET_ERROR;
        else if (received == 0)
            break
        left -= received;
        ptr += received;
    }

    return (len - left);
}
```

```

void __cdecl RecvThread (void * p)
{
    SOCKET sock = (SOCKET)p;
    struct _finddata_t fd;
    long handle;
    char buf[256];
    char buf2[2048];
    char *order[5];
    char *dir[100], *save[300], *save2[300], *f_size[300];
    char *result;
    char t_filename[256], r_filename[256], *plaintext, c;
    int i, level=0, j, size, numtotal=0;

    while(1)
    {

        int recvsize = recv(sock,buf,sizeof(buf),0);
        if(recvsize <= 0)
        {
            printf("접속종료);
            break
        }

        buf[recvsize] = '\0'
        printf("client >> %s\n",buf);
        char *NextContext=NULL;
        result = strtok_s(buf,":",&NextContext);
        i=0;
        while(result !=NULL)
        {
            order[i]=(char *)calloc(PSIZE, sizeof(char));
            strcpy_s(order[i], 50, result);
            result = strtok_s(NULL, ":",&NextContext);
            i++;
        }
    }
}

```

```

if(strcmp("1", order[0])==0)
{
    printf("client >> %s\n",buf);
    dir[level]=(char *)calloc(PSIZE, sizeof(char));
    strcpy_s(dir[level], 50, "c:\\secure\\*.");
    handle = _findfirst(dir[level], &fd);
    i=0;
    while (_findnext(handle, &fd)==0)
    {
        save[i]=(char *)calloc(PSIZE, sizeof(char));
        strcpy_s(save[i], 50, fd.name);
        i++;
    }
    _findclose(handle);
    strcpy(buf2, save[1]);
    strcat(buf2, ":");
    for(j=2; j<i; j++){
        strcat(buf2, save[j]);
        strcat(buf2, ":");}

    int sendsize = send(sock,buf2,sizeof(buf2),0);
    if(sendsize <= 0)
        break
}

else if(strcmp("2", order[0])==0)
{
    printf("client >> %s\n",buf);
    dir[level]=(char *)calloc(PSIZE, sizeof(char));
    strcpy_s(dir[level], 256, "c:\\secure\\");
    strcat_s(dir[level], 256, order[1]);
    strcat_s(dir[level], 256, "\\*.");
    handle = _findfirst(dir[level], &fd);
    i=0;

while (_findnext(handle, &fd)==0)
{

```

```

save[i]=(char *)calloc(PSize, sizeof(char));
save2[i]=(char *)calloc(PSize, sizeof(char));
f_size[i] = (char *)calloc(PSize, sizeof(char));
if (fd.attrib == _A_SUBDIR){ strcpy_s(save2[i], 10, "폴 더); }
else if (fd.attrib == _A_ARCH){ strcpy_s(save2[i], 10, "파 일); }
    else{ strcpy_s(save2[i], 10, "숨김파일);}
    strcpy_s(save[i], 50, fd.name);
    _itoa(fd.size, f_size[i], 10);
    strcat_s(save[i], 256, ".");
    strcat_s(save[i], 256, save2[i]);
    strcat_s(save[i], 256, ".");
    strcat_s(save[i], 256, f_size[i]);
    strcat_s(save[i], 256, ".");
    i++;
}
_findclose(handle);
strcpy(buf2, save[1]);
for(j=2; j<i; j++){
    strcat(buf2, save[j]);
}

int sendsize = send(sock,buf2,sizeof(buf2),0);
if(sendsize <= 0)
    break
}

else if(strcmp("..", order[2])==0)
{
    printf("client >> %s\n",buf);
    dir[level]=(char *)calloc(PSize, sizeof(char));
    strcpy_s(dir[level], 256, "c:\\secure\\");
    strcat_s(dir[level], 256, order[1]);
    strcat_s(dir[level], 256, "\\*.");
    handle = _findfirst(dir[level], &fd);
    i=0;

while (_findnext(handle, &fd)==0)

```

```

{
    save[i]=(char *)calloc(PSize, sizeof(char));
    save2[i]=(char *)calloc(PSize, sizeof(char));
    f_size[i] = (char *)calloc(PSize, sizeof(char));
    if (fd.attrib == _A_SUBDIR){ strcpy_s(save2[i], 10, "폴 더); }
    else if (fd.attrib == _A_ARCH){ strcpy_s(save2[i], 10, "파 일); }
        else{ strcpy_s(save2[i], 10, "숨김파일);}
        strcpy_s(save[i], 50, fd.name);
        _itoa(fd.size, f_size[i], 10);
        strcat_s(save[i], 256, ".");
        strcat_s(save[i], 256, save2[i]);
        strcat_s(save[i], 256, ".");
        strcat_s(save[i], 256, f_size[i]);
        strcat_s(save[i], 256, ".");
        i++;
    }
    _findclose(handle);
    strcpy(buf2, save[1]);
    for(j=2; j<i; j++){
        strcat(buf2, save[j]);
    }

    int sendsize = send(sock,buf2,sizeof(buf2),0);
    if(sendsize <= 0)
        break
}

else if(strcmp("3", order[0])==0)
{
    printf("client >> %s\n",buf);
    dir[level]=(char *)calloc(PSize, sizeof(char));
    strcpy_s(dir[level], 256, "c:\\secure\\");
    strcat_s(dir[level], 256, order[1]);
    strcat_s(dir[level], 256, "\\");
    strcat_s(dir[level], 256, order[2]);
    strcat_s(dir[level], 256, "\\*.");
    handle = _findfirst(dir[level], &fd);
}

```

```

        i=0;

while (_findnext(handle, &fd)==0)
{
    save[i]=(char *)calloc(PSize, sizeof(char));
    save2[i]=(char *)calloc(PSize, sizeof(char));
    f_size[i] = (char *)calloc(PSize, sizeof(char));
    if (fd.attrib == _A_SUBDIR){ strcpy_s(save2[i], 10, "폴   더); }
    else if (fd.attrib == _A_ARCH){ strcpy_s(save2[i], 10, "파   일); }
        else{ strcpy_s(save2[i], 10, "숨김파일);}
        strcpy_s(save[i], 50, fd.name);
        _itoa(fd.size, f_size[i], 10);
        strcat_s(save[i], 256, ":");
        strcat_s(save[i], 256, save2[i]);
        strcat_s(save[i], 256, ":");
        strcat_s(save[i], 256, f_size[i]);
        strcat_s(save[i], 256, ":");
        i++;
    }
    _findclose(handle);
    strcpy(buf2, save[0]);
    for(j=1; j<i; j++){
        strcat(buf2, save[j]);
    }

    int sendsize = send(sock,buf2,sizeof(buf2),0);
    if(sendsize <= 0)
        break
}

else if(strcmp("4", order[0])==0)
{
    printf("client >> %s\n",buf);

    ZeroMemory(r_filename, 256);
    strcpy(r_filename, "c:\\secure\\");

```

```

    strcat(r_filename, order[1]);
    strcat(r_filename, "\\");
    strcat(r_filename, order[2]);

    size = atoi(order[3]);
    char *buffile;
    buffile = (char *)calloc(size, sizeof(char));
    while(1){
        recvsize = recvn(sock, buffile, size, 0);
        if (recvsize == 0)
            break
        numtotal+=recvsize;
    }cout<<r_filename<<endl;

    setlocale(LC_ALL, "korean");
    ofstream fdest(r_filename, ios::out | ios::binary);
    for(i = 0; i < size; i++)
    {
        fdest << buffile[i];
    }
    fdest.close();
    free(buffile);
}

else if(strcmp("5", order[0])==0)
{
    printf("client >> %s\n",buf);
    ZeroMemory(t_filename, 256);
    strcpy(t_filename, "c:\\secure\\");
    strcat(t_filename, order[1]);
    strcat(t_filename, "\\");
    strcat(t_filename, order[2]);
    FILE* fin = fopen(t_filename, "r" );
    fseek( fin, 0, SEEK_END );
    size = ftell( fin );
    ifstream fsrc(t_filename, ios::in | ios::binary);
    fsrc.seekg(0, fsrc.beg);

```

```

        plaintext= (char *)calloc(size, sizeof(char));
        for(i=0; i<size; i++)
        {
            c= fsrc.get();
            cout<<c<<endl;
            plaintext[i]=c;
        }

        int sendsize = send(sock, plaintext, size, 0);
        if(sendsize <= 0)
            break
        cout<<t_filename<<endl;
        fsrc.close();
    }

    else
    {
        int sendsize = send(sock,buf,strlen(buf),0);
        if(sendsize <= 0)
            break
    }
    free(order[0]);
}
closesocket(sock);
}

```

```

int main()
{
    FILE *fp;
    time_t current_time;
    char *date, *log, *ipn, *portn;
    int log_size;

    WSADATA wsaData;
    int retval = WSASStartup(MAKEWORD(2,2),&wsaData);
    if(retval != 0)

```

```

{
    printf("WSAStartup() Error\n");
    return 0;
}

SOCKET serv_sock;
serv_sock = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if(serv_sock == SOCKET_ERROR)
{
    printf("socket() Error\n");
    return 0;
}

SOCKADDR_IN serv_addr = {0};
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(4000);
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);

retval = bind(serv_sock,(SOCKADDR*)&serv_addr,sizeof(SOCKADDR));
if(retval == SOCKET_ERROR)
{
    printf("bind() Error\n");
    return 0;
}

listen(serv_sock,5);
SOCKADDR_IN clnt_addr = {0};
int size = sizeof(SOCKADDR_IN);

while(1)
{
    time(&current_time);
    date=ctime(&current_time);

    SOCKET clnt_sock = accept(serv_sock,(SOCKADDR*)
                             &clnt_addr,&size);
    if(clnt_sock == SOCKET_ERROR)

```

```

{
    printf("accept() Error\n");
    continue
}

ipn=inet_ntoa(clnt_addr.sin_addr);
log_size = strlen("시간: ") + strlen(date) + strlen("IP : ") + strlen(ipn)
           + strlen(" 접속\n");

log=(char *)calloc(log_size, sizeof(char));
strcpy(log, "시간: ");
strcat(log, date);
strcat(log, "IP : ");
strcat(log, ipn);
strcat(log, " 접속\n");
printf("클라이언트접속);
printf("IP : %s, Port : %d\n", inet_ntoa(clnt_addr.sin_addr),
clnt_addr.sin_port);
if((fp = fopen("logfile.txt", "a")) == NULL)
{
    printf("접속로그저장);
    return 1;
}
fwrite(log, log_size, 1 ,fp);
fclose(fp);

_beginthread(RecvThread,NULL,(void*)clnt_sock);

}

closesocket(serv_sock);
WSACleanup();

}

```

```

C:\Windows\system32\cmd.exe
클라이언트 접속
IP : 127.0.0.1, Port : 42487
client >> 1:
client >> 1
접속종료

클라이언트 접속
IP : 127.0.0.1, Port : 43511
client >> 2:박태형:
client >> 2
접속종료

클라이언트 접속
IP : 127.0.0.1, Port : 43767
client >> 2:윤준원:
client >> 2
접속종료

클라이언트 접속
IP : 127.0.0.1, Port : 44023
client >> 2:이희웅:
client >> 2
접속종료

클라이언트 접속
IP : 127.0.0.1, Port : 44279
client >> 2:정세욱:
client >> 2
접속종료

클라이언트 접속
IP : 127.0.0.1, Port : 44535
client >> 2:정호윤:
client >> 2
접속종료

```

```

logfile.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도
IP : 127.0.0.1 접속
시간 : Wed May 18 00:51:43 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 00:51:44 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 00:51:46 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 00:51:47 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 00:52:07 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:18 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:27 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:31 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:32 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:33 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:49:33 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:07 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:16 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:20 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:21 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:22 2016
IP : 127.0.0.1 접속
시간 : Wed May 18 02:50:22 2016
IP : 127.0.0.1 접속

```

〈서버 접속자 현황 및 로그 기록〉

4. 결론

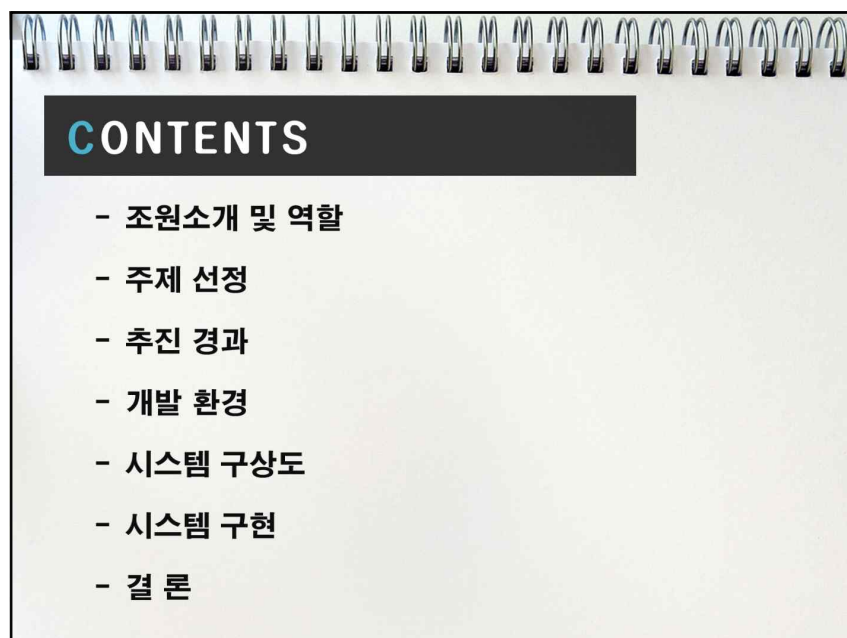
본 프로그램을 기업에서 사용할 경우 접근에 제한이 있는 장소에 시스템을 설치하여 파일을 암호화하여 저장한다면 인가된 사용자만이 해당 시스템에 접근 복호화 할 수 있기 때문에 외부 요인으로 인한 보안피해를 방지 할 수 있다.

개인의 시스템에서 사용할 경우 중요한 파일들을 암호화하여 저장해둔다면 파일이 유출이 된다 하더라도 직접 개인의 시스템에 접근하지 않으면 복호화 할 수 없기 때문에 일차적으로 보안피해를 방지 할 수 있다.

5. 참고 자료

- W.Stalling. 1999. Cryptography and Network Security. Prentice Hall.
- 박해원, 신경욱. "64비트 블록암호 알고리즘 HIGHT의 효율적인 하드웨어 구현." 한국해양정보통신학회논문지, v.15, no.9, pp.1993-1999. 2011.
- 정보통신단체표준, TTAK.KO-12.0040/R1, 2008.
- 핵심 API로 배우는 윈도우 프로그래밍

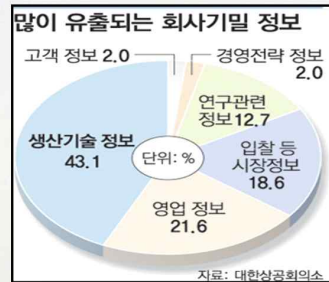
6. 발표 자료



조원소개 및 역할

이름		담당역할
조장	윤준원	프로젝트 총괄 및 서버 프로그램 구현
조원	정호운	API 프로그래밍
조원	박태형	암호화 기능 구현
조원	이희웅	키 암호화 기능 구현
조원	정세욱	GUI 기능 구현

주제 선정



해킹 정보유출 사례

사업자명	발생일자	개인정보 유출 건수
SK컴즈	2011년 7월	3500만
넥슨	2011년 11월	1320만
KT	2011년 7월	873만
EBS	2011년 5월	420만
현대캐피탈	2011년 4월	175만
엘손	2011년 8월	35만

- 현재 많은 기업에서 정보유출 사례가 일어나고 있음.
- 중소기업이나 학교와 같은 소규모 네트워크에서도 일어나고 있음.
- 소규모 환경에서 파일을 안전하게 저장 및 전달 할 수 있는 서버 프로그램을 구축하고자 함.

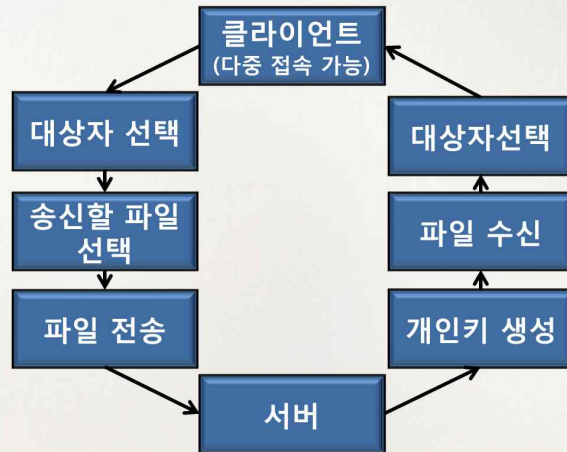
추진 경과

추진 일정	1월	2월	3월	4월	5월
GUI 구현					
클라이언트 구현					
서버 프로그램 구현					
암호화 기능 구현					
프로그램 점검 및 개선사항 개선					

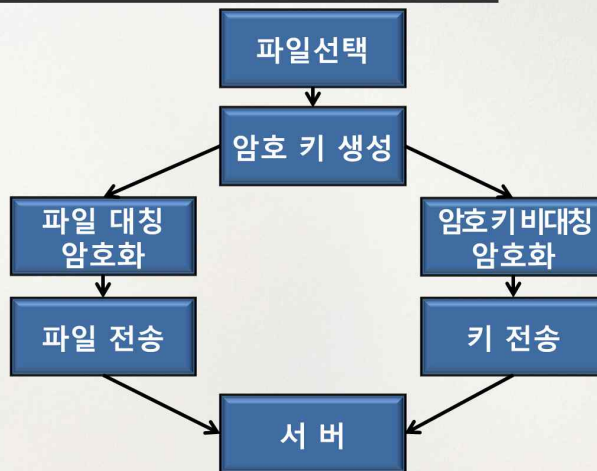
개발 환경

- 개발언어 ⇨ Visual 2008 C++
- 사용 소스 ⇨ API, OPEN SSL, 소켓 프로그래밍, RSA, HIGHT 암호
- 운영체제 ⇨ Microsoft Windows 7

시스템 구상도(클라이언트)



시스템 구상도(암호화 방식)



주요개발 사항(1/4)

➤ 하이트(HIGHT) 암호 알고리즘을 통한 파일 암호화

- 파일 암호화에 사용
- HIGHT는 초경량 블록암호 알고리즘
- 기본적인 산술 연산들인 XOR, 덧셈, 순환이동
- SEED, AES 등 기타 알고리즘보다 간단한 알고리즘 구조

주요개발 사항(2/4)

➤ RSA 암호 알고리즘을 통한 비대칭키 암호화

- 암호키값의 암호화에 사용
- RSA는 비대칭키 알고리즘
- 기존의 암호화 방식과 다르게 개인키와 공개키가 존재
- 암호화에는 공개키를 사용하고 복호화에는 개인키를 사용

주요개발 사항 (3/4)

➤ API를 통한 GUI 구현

- C/C++ 프로그램에서 직접 운영 체제와 상호 작용
- 단추와 창, 글자 상자, 그 외 GUI 개체를 만들
- 운영체제가 응용 프로그램을 위해 제공하는 함수의 집합

주요개발 사항 (4/4)

➤ 소켓(Socket) 프로그래밍을 통한 파일 송·수신

- 소켓(Socket)이란 어플리케이션에게 네트워크 접속을 위한 연결장치, 인터페이스 역할을 함.
- 네트워크 어플리케이션이 보낸 데이터를 소켓을 거쳐 운영체제상에 존재하는 TCP/IP 소프트웨어에게 전달함.

시스템 구현



〈클라이언트, 서버〉

결론

➤ 기대 효과

- HIGHT 암호화 기법을 이용하여 파일 암호화 시간 최소화
- 서버에 파일 암호화 되어서 저장 되기 때문에 외부 요인으로 인한 보안 피해를 최소화
- 인가된 사용자만이 저장된 파일을 복호화 할 수 있음
- 암호키값을 비대칭 암호를 통해 다시 암호화 하여 저장 함으로 안전하게 암호키 전달 가능

