2015년 정보보호학과 졸업작품 보고서

USB를 이용한 System Lock 및 File Security Service

팀 명 : Team IU (Intelligent USB)

지도교수 : 양 정 모 교수님

조장 : 조 상 일
유 승 우
조 한 슬

2015. 6
중부대학교 정보보호학과

목 차

Ⅰ. 서 론

Ⅰ-1. 프로젝트 소개
Ⅰ-2. 연구 활동 배경
Ⅰ-3. 피해 사례 조사

Ⅱ. 본 론

Ⅱ-1. 프로그램 구성
Ⅱ-2. 파일의 암호화 후 백업 및 복호화 후 복원
Ⅱ-3. 화면 잠금
Ⅱ-4. 소스 코드

Ⅲ. 결 론

Ⅲ-1. 기대효과
Ⅲ-2. 프로그램의 향후 발전, 개선 방향

Ⅳ. 참고자료

Ⅳ-1. 참고 알고리즘, 블로그

Ⅴ. 별 첨

Ⅴ-1. 발표 PPT

Ⅰ. 서　　론

Ⅰ-1. 프로젝트 소개

1. 프로젝트 제목

USB를 이용한 System Lock 및 File Security Service

2. 프로젝트 선정 사유

최근 경량 노트북의 보급이 급증되었으나 많은 사용자들은 비밀번호를 이용한 화면 잠금 등 보안 기능을 적절히 사용하지 않아 각종 파일에 기록되어있는 개인 정보가 유출될 수 있는 위험이 항상 존재하고 있습니다.

이외에도 누구나 쉽게 구할 수 있고 대중적으로 사용되는 USB의 경우 휴대가 편리하면서 대용량인 경우가 많아 많은 사람이 사용하고 있지만 이 USB가 분실될 경우, 내장되어 있는 중요한 정보의 백업이나 정보의 유출에 따른 피해는 누구도 보상할 수 없는 현실이기 때문에 이러한 문제점들을 해결하기 위하여 본 프로젝트를 선정하였습니다.

3. 프로젝트 연구내용

Visual C를 기반으로 하는 Window32 API를 사용하여 화면 잠금과 특정 파일의 암호화를 구현하였고, C++ 전용 AES128 암호 알고리즘을 C에 맞추어 변형시켜서 적용하였습니다.

4. 프로젝트 요약

저희는 위의 두 가지 문제점이라는 토끼를 한 번에 잡기 위하여 USB를 이용하여 버튼 하나로 실행되는 강력한 화면 잠금 기능을 구현하여 PC나 노트북 등에 저장된 정보를 1차적으로 보호하여 미리 저장해둔 Hash값이 서로 Matching되는 USB를 소지한 사용자만이 화면 잠금을 해제할 수 있도록 구현함과 동시에 USB에 저장된 중요한 파일의 경우는 사용자의 과실로 인한 분실에 따른 개인정보의 유출을 방지하기 위하여 파일의 내용을 암호화하여 알아볼 수 없게 하는 기능과 적용된 암호를 복호화가 가능하도록 구현하였습니다.

Ⅰ-2. 연구 활동 배경

- 현대 사회를 살아감에 있어서 정보의 의미는 개인이나 특정 기업 혹은 단체와 국가에 이르기까지 그 필요성과 중요성을 널리 인식하고 있는 반면 아직 그 정보를 보호하고자 하는 개개인의 노력은 아직도 부족하고 미흡한 편이라고 감히 말할 수 있습니다.

이에 저희 Team I.U는 소형 USB 하나로 2가지의 기본적인 보안 기능을 제공하는 연구를 시작하였습니다. 본 연구 활동의 결과로 만들어진 프로그램을 적절히 사용할 경우 개인적인 정보 뿐 만 아니라 기업, 단체 또는 국가의 정보를 개인이 관리함에 있어서 한층 더 강화된 보안 기능을 제공받을 수 있습니다.

본 연구 활동의 주요 기능은 크게 화면 잠금과 암호화 기능으로 나눌 수 있습니다.

우선 화면 잠금 기능을 구현하기 위해 “Windows32 API”를 사용하였는데, 이 윈도 API는 마이크로소프트 32비트 윈도 운영 체제들이 사용하는 API입니다. C/C++ 개발 프로그램에서 직접 운영 체제와 상호 작용할 수 있도록 만들어졌으며, 이 API에 소속된 DLL(Dynamic Link Library) 중 “WinLockDLL”이란 라이브러리를 사용하여 구현하였습니다.

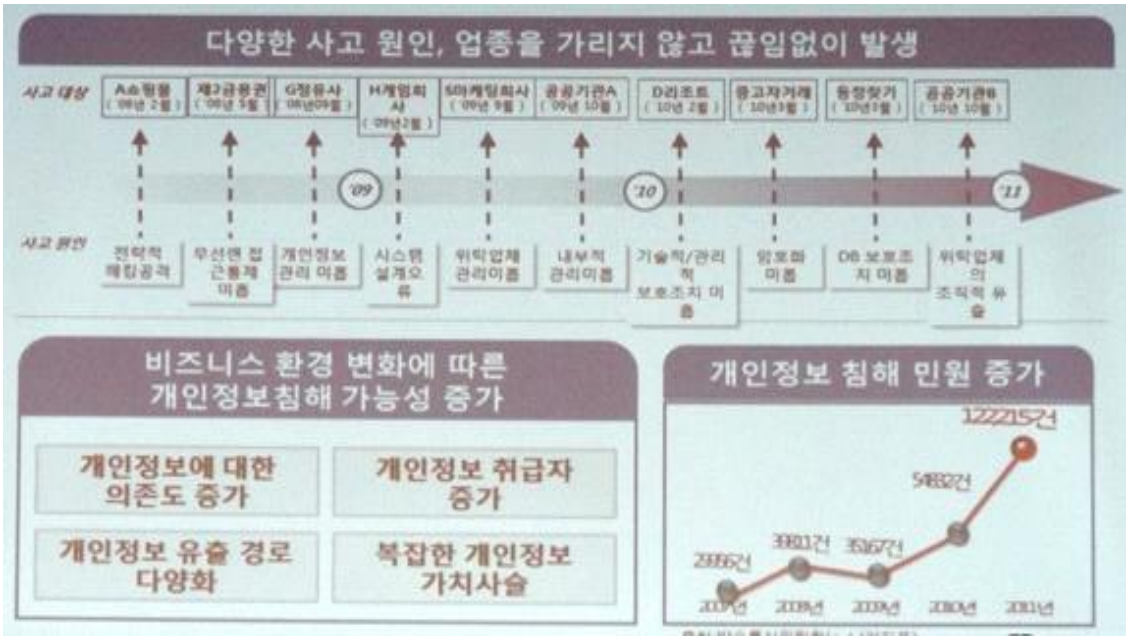
암호화 기능은 현재 Open Source로 공개되어있는 AES 암호를 사용하였습니다. 이 AES 암호란 Advanced Encryption Standard의 약자로, 존 대먼, 빈센트 라이먼이 만든 암호화 알고리즘입니다. NIST가 제정하였고 미국 정부가 공식적으로 채택하게 된 후 널리 사용되고 있습니다. 128Bit, 192Bit, 256Bit 3가지 종류를 지원하는 암호 알고리즘이지만 본 연구에서는 속도의 최적화를 위해 128Bit 방식을 사용하였습니다.

특히 이 암호화 기능의 강도를 유지하기 위해 사용하는 해시 함수(Hash Function)를 사용하는데 이는 임의의 길이 데이터를 일정한 길이의 데이터로 Mapping시키는 알고리즘입니다. 해시 함수는 결정론적으로 작동해야 하며, 따라서 두 해시 값이 다르다면 그 해시값에 대한 원래 데이터도 달라야 하며, 역은 성립하지 않는 특징이 있습니다. 해시함수 중에는 암호학적 해시함수(Cryptographic Hash Function)와 비-암호학적 해시함수로 구분되곤 합니다.

저희는 이 Hash 함수 중 SHA-1 함수를 사용하였습니다. SHA(Secure Hash Algorithm) 함수들은 서로 관련된 암호학적 해시 함수들의 모음입니다. 이들 함수는 미국 국가안보국(NSA)이 1993년에 처음으로 설계했으며 미국 국가 표준으로 지정되었던 함수입니다. 기존의 SHA-1 이외에도 4종류의 변형(SHA-224, SHA-256, SHA-384, SHA-512)가 발표되었으며, 이들을 통칭해서 SHA-2라고 하기 도 합니다.

SHA-1은 SHA 함수들 중 가장 많이 쓰이며, TLS, SSL, SSH, IPSec 등 많은 보안 기능을 지원하는 프로토콜과 프로그램에서 사용되고 있습니다. 본 연구에서는 속도의 최적화를 위해 비교적 간단한 SHA-1을 사용하였습니다.

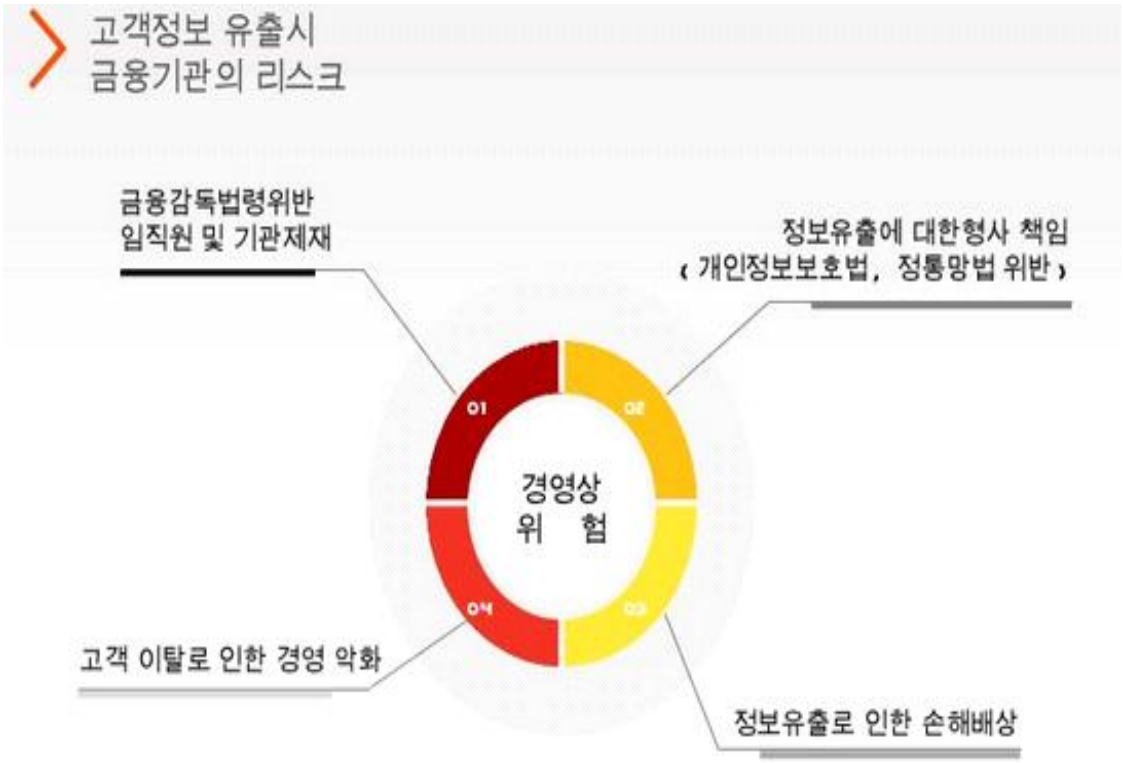
Ⅰ-3. 피해 사례 조사



<그림-1.3(1) 개인 정보 침해 관련 사건 자료>

위의 통계에서 확인할 수 있는 점은 사고 대상의 업종과는 관계없이 개인 정보의 관리 미흡으로 인한 유출 사태가 끊임없이 일어나고 그 추세 또한 급격히 상승하고 있다는 것을 확인할 수 있습니다.

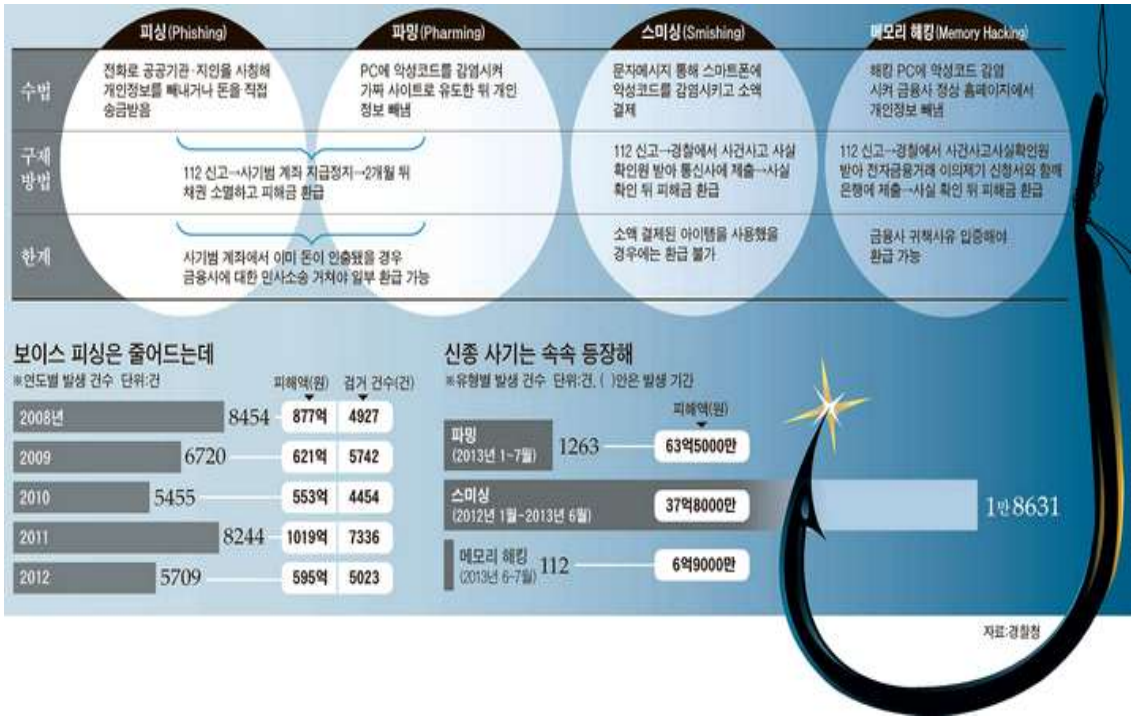
위와 같은 피해 사례 중 개인의 부주의 등 인적 과실로 인한 피해는 작은 경각심과 보안 의식을 갖고 있다면 누구나 예방할 수 있을 것이라고 생각되어 연구를 진행하게 되었습니다.



<그림-1.3(2) 고객 정보 유출 시 금융기관이 입는 피해>

위의 자료는 고객의 정보 유출 피해가 발생했을 때 기업이 직접적으로 타격을 입을 수 있는 부분입니다. 일례로 SK communications의 NateON과 Cyworld 서비스의 경우 대량의 개인정보 유출 사건 당시 고객들은 SK를 상대로 집단 소송을 이행한 바 있으며, 사용자의 대규모 이탈의 직접적인 원인이 되어 현재는 그 명맥만 잇고 있는 현실입니다.

또 다른 예로는 농협의 대량 개인 정보 유출 사건 당시 많은 고객들이 신뢰를 잃고 농협과의 모든 금융 거래를 중지하고 신용 정보를 파기한 후 타사 은행의 고객으로 유입된 바가 있습니다.



<그림-1.3(3) 개인정보 유출 사고 시 구제 방법 및 한계>

위의 자료에서 확인할 수 있듯 해커의 악의적인 수법으로 인한 사용자의 개인 정보 유출이 일어났을 때에는 해당 기업에게 보상을 청구할 수 있는 법적 근거가 마련되어 있지만, 만약 직원이나 내부 관계자의 부주의로 인한 사고로 인한 보상은 어디에도 나타나 있지 않습니다.

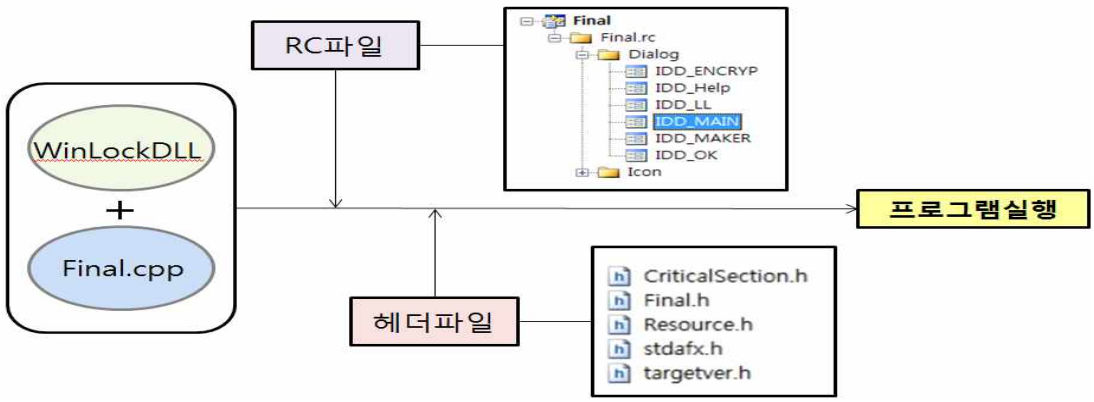
모든 책임을 특정인에게 전가하기에는 기업이나 사고의 원인이 된 당사자 측 모두 부담이 뒤따르기 때문입니다.

저희는 본 연구를 통해 제작된 프로그램으로 위와 같은 사건, 사고를 사전에 미리 예방하고자 하여 실시하게 되었습니다.

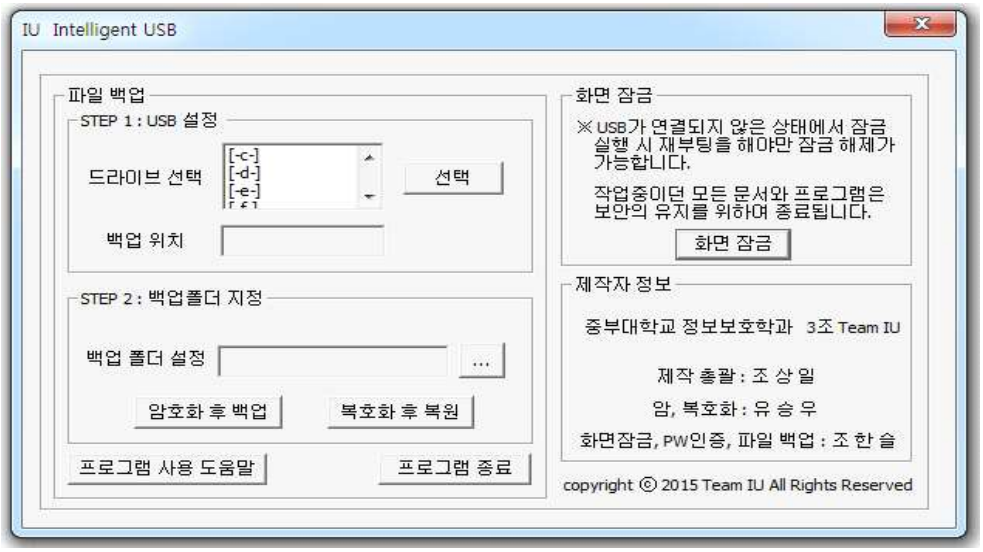
Ⅱ. 본　　론

Ⅱ-1. 프로그램 구성

- USB 인식 후 잠금 해제를 위한 PW 파일 확인
- WinLockDLL을 이용한 화면 잠금
- PC(노트북) ↔ USB간의 파일 암호화 백업, 복호화 복원



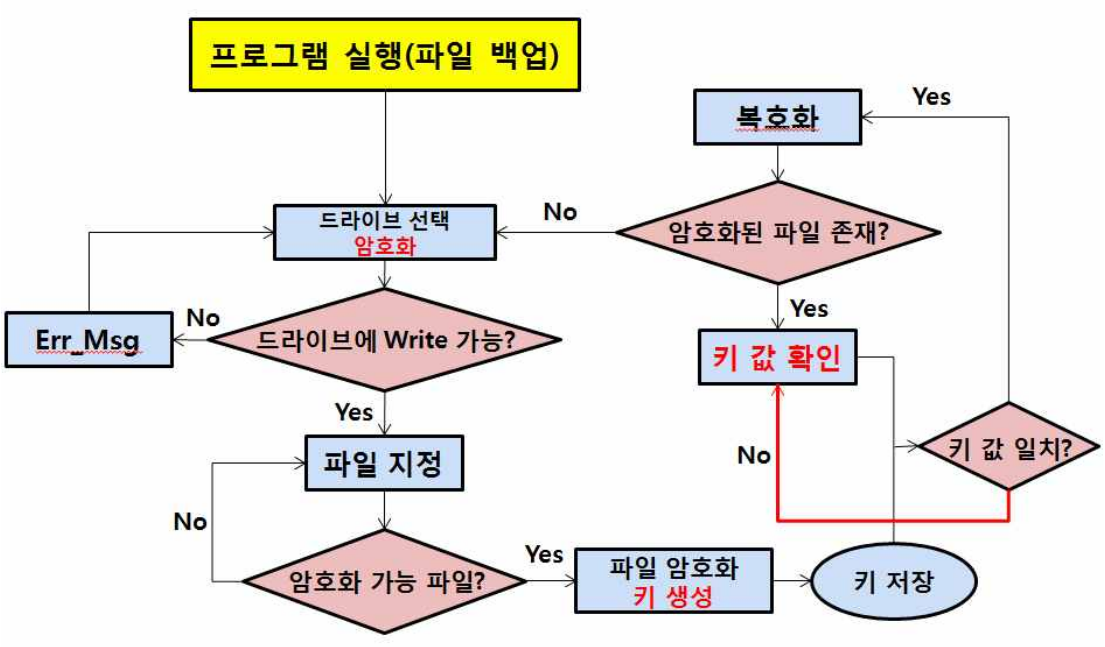
< 그림-2.1(1) 프로그램 구조 >



< 그림-2.1(2) 프로그램 메인 화면 >

Ⅱ-2. 파일의 암호화 후 백업 및 복호화 후 복원

파일 백업 Flow Chart



< 그림-2.2 파일 백업 Flow Chart >

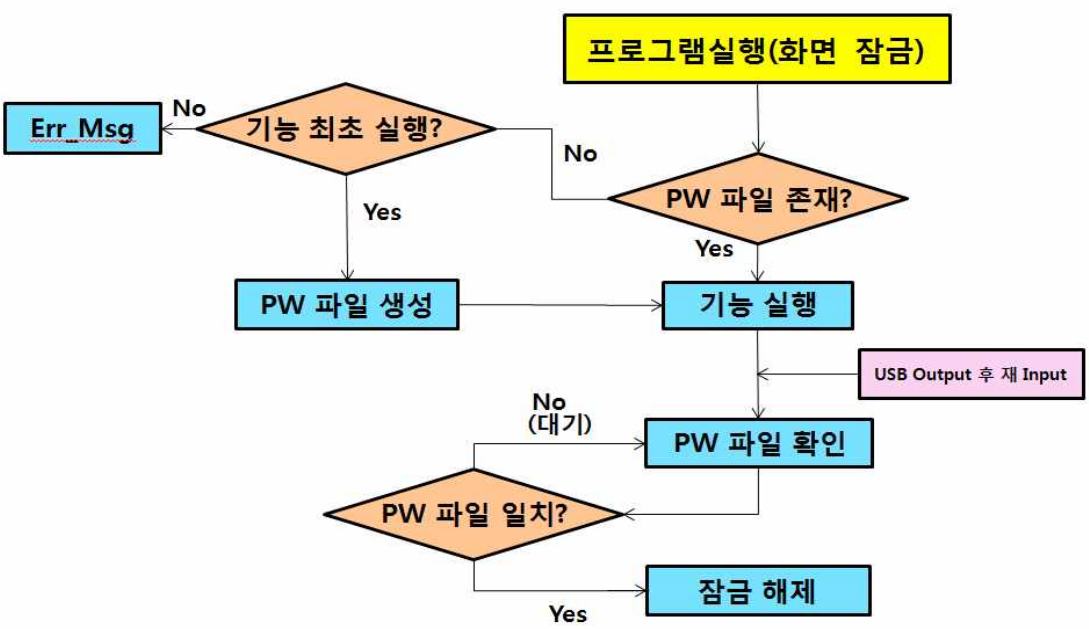
파일의 백업 기능은 기존의 Copy & Paste를 이용하는 수동 방식을 개선한 형태로 폴더를 지정한 뒤 클릭 한 번으로 백업 및 암호화 기능이 실행되게 구축하여 인식된 상태인 USB의 최상위 Directory에 “Backup” 이라는 이름의 폴더를 생성한 후 AES-128 암호를 적용하여 저장하게 되고, 복호화를 선택할 경우 기존에 백업을 시도했던 경로에 동일한 방식의 암호를 역산하는 방식으로 복호화 한 뒤 저장합니다.

기능의 동작 순서는 프로그램을 실행 후 메인 화면에서 “드라이브 선택” 메뉴를 통한 백업 드라이브(USB)를 선택 후 Write가 가능한지 여부를 판단하게 됩니다. 만약 Write가 가능한 드라이브라면 백업할 파일을 지정한 후 암호화가 가능한 파일 형식인지의 여부를 확인한 후에 백업과 동시에 파일 암호화 키를 생성하게 되고 키를 저장합니다.

복호화 시에는 생성되었던 키 값에 Hash 함수를 적용한 결과값이 USB에 저장된 것과 PC에 저장되어있는 것이 일치하는 지 여부를 확인 후 일치한다면 복호화가 이루어지게 됩니다.
모든 선택문에서 False 값으로 판단될 경우 이전 단계로 돌아가 메뉴를 재 선택하게 됩니다.

Ⅱ-3. 화면 잠금

화면 잠금 Flow Chart



< 그림-2.3 화면 잠금 Flow Chart >

본 프로그램이 실행된 적이 없는 USB를 PC나 노트북에 삽입하여 정상적으로 인식된 경우 프로그램을 실행한 후 파일 백업 메뉴의 Step 1에서 보여지는 드라이브를 선택하는 화면에서 인식된 드라이브(현재 구동중인 PC의 경우 G:\)에 LockKey.dat 이라는 파일을 숨김 형태로 생성하여 Hash 함수인 SHA-1를 적용시킨 PW파일로 지정 후 화면 잠금 해제 시도가 있을 때(USB Output 후 다시 Input 시) 매칭을 확인하여 잠금을 해제할 수 있습니다.

기능의 동작 순서는 메인 화면에서 화면 잠금 해제의 키 값을 저장할 USB 드라이버를 선택 후 우측 상단 “화면 잠금” 버튼을 클릭 시 화면이 잠금 설정되게 됩니다. 만약 화면 잠금을 해제할 키를 USB에 저장하지 않고 기능을 실행 시 잠금 해제가 불가능하기 때문에 사용중인 PC를 재부팅 하여야 합니다.

USB 삽입 시 저장된 화면 잠금 해제용 키 값이 정상적인 PW파일로 인식되면 화면 잠금을 해제하게 됩니다.

모든 선택문에서 False 값으로 판단될 경우 이전 단계로 돌아가 대기하게 됩니다.

Ⅱ-4. 소스 코드

```
#define       WIN32_LEAN_AND_MEAN
#include "stdafx.h"
#include "resource.h"---
#include "Logfile.h"
#include "../../WinLockDLL/WinLockDLL/winlockdll.h"
#include "Shlwapi.h"
#include <dbt.h>
#include <windows.h>//파일복사
#include <shlwapi.h>//LockKey.txt 파일검사
#include <WinCrypt.h>//파일암호화

#pragma comment(lib, "Shlwapi.lib")
#define       PROGRAM_MANAGER "Program Manager"        // Program manager window name
#define       TASKBAR           "Shell_TrayWnd"              // Taskbar class name



//#define     MY_PASS         "NALNARI"            // 패스워드//파일암호화
#define     KeyLen          0x0080 * 0x10000     // 128-bit //파일암호화

HHOOK               hHook;            // Mouse hook
HINSTANCE           hInst;                  // Instance handle
HINSTANCE            g_hInstance;
HWND hWnd ;

//파일생성및쓰기
static      HANDLE  hFile;
static DWORD        result;

static int              GStr;
//파일읽기
static   HANDLE R_hFile;
static   HANDLE f_hFile;
static   DWORD   RRead;
static   DWORD   FRead;
static   TCHAR   Rstring[21];
static TCHAR   fstring[21];
static int      RStr;
static DWORD ppbData;//암호화
//파일 생성 및 검사
TCHAR Txtname[128]={TEXT(":\\LockKey.dat")};//파일이름
TCHAR CTxtname[128]={TEXT("D")};
TCHAR Tempname[128];
TCHAR *fTxtname[128]={TEXT("e"),TEXT("f"),TEXT("g"),TEXT("h"),TEXT("i"),TEXT("j")};
TCHAR copyTxtname[128];
```

```
//LPCTSTR lpszClass=TEXT("Screen Lock");
//파일 내용 암호화
HCRYPTPROV      hProv;
HCRYPTHASH      hHash;
HCRYPTKEY       hKey;
BYTE            pbData[100] = "TSTWPSELSIDWLSEP"
BYTE            MY_PASSKEY;
BYTE                    MY_PASSWD[100];



DWORD           dwDataLen = (DWORD)strlen((char*)pbData) + 1;
//폴더 복사
TCHAR               FolderCP[128];
TCHAR               FolderCP2[128];
TCHAR               NUL[1];
#define DESKTOPNAME "MyDesktop2"    // New desktop name
THREAD_DATA td;
static WCHAR buf[1024];
static char cbuf[1024];
bool        g_bUnlock = false


// 파일 암호화
// wstrSrcFN : 원시파일명
// wstrDestFN : 목적파일명
// hKey : 암호화키


HRESULT FileEncryption( WCHAR *wstrSrcFN, WCHAR *wstrDestFN, HCRYPTKEY hKey )
{
        if( wstrSrcFN == NULL ||
                wstrDestFN == NULL ||
                hKey == NULL )
                return S_FALSE;

        HANDLE  hSrcFile, hDestFile;
        DWORD   nNumberOfBytesToRead, lpNumberOfBytesRead;
        DWORD   lpNumberOfBytesWritten;
        char        strBuffer[1024];

                // 원시파일 오픈
        hSrcFile = CreateFile( wstrSrcFN, GENERIC_READ, 0, NULL, OPEN_EXISTING,
                                FILE_ATTRIBUTE_NORMAL, NULL );
        if( hSrcFile == INVALID_HANDLE_VALUE )
                return S_FALSE;
                // 목적파일 오픈
        hDestFile = CreateFile( wstrDestFN, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                                FILE_ATTRIBUTE_NORMAL, NULL );

        nNumberOfBytesToRead = sizeof(strBuffer);
```

```
        while(1)
        {
                        // 원시파일로부터 일정량(nNumberOfBytesToRead)의 데이터를 읽는다.
                ReadFile( hSrcFile, strBuffer, nNumberOfBytesToRead, &lpNumberOfBytesRead, NULL
);
                        // 읽어들인 데이터가 없을 경우 종료
                if( lpNumberOfBytesRead <= 0 )
                        break
                        // 암호화
                CryptEncrypt( hKey, 0, TRUE, 0, (BYTE *)strBuffer, &lpNumberOfBytesRead,
                        sizeof(strBuffer) );
                        // 암호화처리된데이터를목적파일에저장
                WriteFile( hDestFile, strBuffer, lpNumberOfBytesRead, &lpNumberOfBytesWritten,
                        NULL );
        }
        CloseHandle( hDestFile );       // 오픈된 목적파일 핸들 닫기
        CloseHandle( hSrcFile );        // 오픈된 원시파일 핸들 닫기
        return S_OK;
}


// 파일 복호화
// wstrSrcFN : 원시파일명
// wstrDestFN : 목적파일명
// hKey : 복호화키
HRESULT FileDecryption( WCHAR *wstrSrcFN, WCHAR *wstrDestFN, HCRYPTKEY hKey )
{
        if( wstrSrcFN == NULL ||
                wstrDestFN == NULL ||
                hKey == NULL )
                return S_FALSE;
        HANDLE  hSrcFile, hDestFile;
        DWORD   nNumberOfBytesToRead, lpNumberOfBytesRead;
        DWORD   lpNumberOfBytesWritten;

        char    strBuffer[1024];
                // 원시파일 오픈
        hSrcFile = CreateFile( wstrSrcFN, GENERIC_READ, 0, NULL, OPEN_EXISTING,
                                FILE_ATTRIBUTE_NORMAL, NULL );
        if( hSrcFile == INVALID_HANDLE_VALUE )
                return S_FALSE;
                // 목적파일 오픈
        hDestFile = CreateFile( wstrDestFN, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                                FILE_ATTRIBUTE_NORMAL, NULL );

        if( hDestFile == INVALID_HANDLE_VALUE )
        {
                CloseHandle( hSrcFile );        // 오픈된 원시파일 핸들 닫기
                return S_FALSE;
        }
        nNumberOfBytesToRead = sizeof(strBuffer);
```

```
		while(1)
		{
				// 원시파일로부터 일정량(nNumberOfBytesToRead)의 데이터를 읽는다.
			ReadFile( hSrcFile, strBuffer, nNumberOfBytesToRead, &lpNumberOfBytesRead, NULL
);
				// 읽어들인 데이터가 없을 경우 종료
			if( lpNumberOfBytesRead <= 0 )
				break
				// 복호화
			CryptDecrypt( hKey, 0, TRUE, 0, (BYTE *)strBuffer, &lpNumberOfBytesRead );
				// 암호화 처리된 데이터를 목적 파일에 저장
			WriteFile( hDestFile, strBuffer, lpNumberOfBytesRead, &lpNumberOfBytesWritten,
				NULL );
		}
		CloseHandle( hDestFile );	// 오픈된 목적파일 핸들 닫기
		CloseHandle( hSrcFile );	// 오픈된 원시파일 핸들 닫기
		return S_OK;
}

//LRESULT CALLBACK WndProc(HWND hWnd,UINT iMessage,WPARAM wParam,LPARAM lParam);
INT_PTR CALLBACK MainDlgProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);
INT_PTR CALLBACK UnlockEditProc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam);
BOOL BrowseFolder(HWND hParent, LPCTSTR szTitle, LPCTSTR StartPath, TCHAR *szFolder);

char Remove(char* in_hasGenre);

BOOL XCopy(LPCTSTR Src, LPCTSTR Dest);
//BOOL FileEncryption(LPCTSTR Src,LPCTSTR Dest);//, LPCTSTR Dest);
BOOL FileEncryption(LPCTSTR Src, LPCTSTR Dest,HWND hwnd);//, LPCTSTR Dest) // Foldercp
FolderCP2가 들어와야함
BOOL FileDecryption(LPCTSTR Src, LPCTSTR Dest,HWND hwnd);//, LPCTSTR Dest) // Foldercp
FolderCP2가 들어와야함

INT_PTR CALLBACK HelpProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam);
INT_PTR CALLBACK MakerProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam);
INT_PTR CALLBACK OkProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam);
//void hex2byte(const char *in, int len, byte *out);//파일암호화
//void encryp();


DWORD WINAPI MyThread(LPVOID lpParameter){
		SetThreadDesktop(((THREAD_DATA *)lpParameter)->hDesk);
		g_bUnlock = false
		DialogBox( g_hInstance, MAKEINTRESOURCE(IDD_LL), HWND_DESKTOP, UnlockEditProc );
		while( !g_bUnlock ){
			Sleep(10);
		}
		return 0;
}
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow){

	//HWND hWnd;
	//MSG Message;
	//WNDCLASS WndClass;
	g_hInstance = hInstance;
	DialogBox( g_hInstance, MAKEINTRESOURCE(IDD_MAIN), HWND_DESKTOP, MainDlgProc );
return 0;
}

int i=0,j=0;
HWND hEdit;
INT_PTR CALLBACK MainDlgProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam){
//RECT  r;

TCHAR usblist[128];//암호키 생성
static TCHAR StartPath[MAX_PATH];
TCHAR Folder[MAX_PATH];
TCHAR File[MAX_PATH]={TEXT("*.")};//리스트박스
char filetest[MAX_PATH];//([--]제거할때)형변환

	switch(Message){
		case WM_CREATE:
			HWND hCombo;
			hCombo=GetDlgItem(hDlg, IDOK);
			SetFocus(hCombo);
		return TRUE;

		//USB 선택 리스트박스
		case WM_INITDIALOG:
			DlgDirList(hDlg, File, IDC_LIST2, 0, (DDL_DRIVES));

			return TRUE;


		//USB 인식부분
		case WM_DEVICECHANGE:
				switch (LOWORD(wParam))
				{
					case DBT_DEVICEARRIVAL: DlgDirList(hDlg, File,
						IDC_LIST2, 0, (DDL_DRIVES));
						break
					case DBT_DEVICEREMOVECOMPLETE: DlgDirList(hDlg,
						File, IDC_LIST2, 0, (DDL_DRIVES));
						break
					default:
						break
				}
				return TRUE;
```

```
                case WM_COMMAND:
                        switch (LOWORD(wParam))
                        {
                                //화면 잠금 버튼
                                case IDOK: MessageBox(hWnd, TEXT("USB를분리해주세요"),
                                                        TEXT("확인"),MB_OK);
                                        strcpy(td.szDesktopName, DESKTOPNAME);
                                        Thread_Desktop(MyThread, (THREAD_DATA *)&td);
                                        return TRUE;


//암호키 생성(USB선택완료)버튼

                                case IDC_ENCRYPTION:
                                        HWND hList;
                                        hList=GetDlgItem(hDlg, IDC_LIST2);
                                        i=SendMessage(hList, LB_GETCURSEL,0,0);
                                        SendMessage(hList, LB_GETTEXT, i, (LPARAM)File);
                                if(i<0)
                                        MessageBox(hWnd,TEXT("usb를선택해주세요"),TEXT("경고"),MB_OK);
                                else if(i==0||i==1) MessageBox(hWnd,TEXT("C와 D드라이브를 제외한 usb를 선택
                                                                        해주세요"),TEXT("경고"),MB_OK);
                                else
                                {
                                    WideCharToMultiByte(CP_ACP, 0, File, 260, filetest, 260, NULL, NULL);
                                        Remove(filetest);
                                        MultiByteToWideChar( CP_ACP, 0, filetest, -1, File, 260);
                                        wcscpy(FolderCP2, NUL);
                                        wcscat(FolderCP2, File);
                                        wcscat(FolderCP2, L":\\BackUp");
                                        //MessageBox(hWnd, FolderCP2,TEXT("알림"),MB_OK);
SetDlgItemText(hDlg, IDC_EDIT2, FolderCP2);


                                        wcscat(File, (LPCTSTR)Txtname);
                                        hFile = CreateFile(File, GENERIC_WRITE, FILE_SHARE_WRITE, 0,
                                                CREATE_ALWAYS, FILE_ATTRIBUTE_HIDDEN, 0);
                                //CREATE_ALWAYS 를OPEN_EXISTING 로 수정하면 파일 끝에 추가

//암호화 부분
                        if(!CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0))
//CSP(Crystographic Service Provider) 핸들얻기
                        CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV,PROV_RSA_FULL,
                                        CRYPT_NEWKEYSET);
// 유저용 키 컨테이너 만들기

CryptGenRandom(hProv,1,&MY_PASSKEY); // 난수생성

wsprintf((LPWSTR)MY_PASSWD,TEXT("%d273%d58%d"),MY_PASSKEY,MY_PASSKEY,MY_PASSKEY);
//MessageBox(hWnd, (LPWSTR)MY_PASSWD,TEXT("MY_PASSKEY"),MB_OK);
```

```
CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);// 해쉬 만들기
CryptHashData(hHash, (BYTE*)MY_PASSWD,sizeof(MY_PASSWD) ,0);
//(DWORD)strlen(MY_PASSWD), 0);  // 해쉬값 계산
CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128
CryptEncrypt(hKey, 0, TRUE, 0, pbData, &dwDataLen, 30);//암호화
WriteFile(hFile, pbData, sizeof(pbData) ,&result, NULL);
//CryptDecrypt(hKey, 0, TRUE, 0, pbData, &dwDataLen);//복호화

CryptDestroyKey(hKey);
CryptDestroyHash(hHash); // 해쉬 없애기
CryptReleaseContext(hProv, 0); // CSP 핸들풀어주기

CloseHandle(hFile);
wcscpy(Tempname,CTxtname);
wcscat(Tempname, (LPCTSTR)Txtname); //d드라이브에 파일 복사

CopyFile(File,Tempname,false);    //d드라이브에 파일 복사
//wsprintf(usblist,TEXT("%s 파일생성완료"),File);
//MessageBox(hWnd, usblist,TEXT("알림"),MB_OK);
        }
return TRUE;
                //파일 암호화 버튼
                case IDC_FILE_ENCRYPTION:
                //MessageBox(hWnd, TEXT("파일암호화"),TEXT("메시지박스"),MB_OK);
                //복사하는 기능 및 암호화 기능
                GetDlgItemText(hDlg, IDC_EDIT,FolderCP,150);
                if (!lstrlen (FolderCP) )
                        MessageBox(hWnd,L"백업폴더를지정해주세요!",L"알림",MB_OK);
                else
                {
                        GetDlgItemText(hDlg, IDC_EDIT2,FolderCP2,150);
                        if (!lstrlen (FolderCP2) )
                                MessageBox(hWnd,L"USB를지정해주세요!",L"알림",MB_OK);
                        else
                        {
                                CreateDirectory(FolderCP2, NULL);
                                if(XCopy(FolderCP, FolderCP2))
                                        MessageBox(hWnd,L"복사를완료했습니다",L"알림",MB_OK);
                                if(FileEncryption(FolderCP,FolderCP2,hDlg))
                                        MessageBox(hWnd,L"진행중",L"FileEncryption",MB_OK);
                        }
                }
                return TRUE;
                //파일 복호화 버튼
                case IDC_FILE_DECRYPTION:
                        CreateDirectory(FolderCP, NULL);
                                if(XCopy(FolderCP2, FolderCP))
                                        MessageBox(hWnd,L"복사를완료했습니다",L"알림",MB_OK);
                                if(FileDecryption(FolderCP2,FolderCP,hDlg))
                                        MessageBox(hWnd,L"진행중",L"FileDecryption",MB_OK);
                return TRUE;
```

```
                    //... 버튼
                    case IDC_FINDFORDER:
                              if (BrowseFolder(hWnd,TEXT("폴더를선택하세요"),StartPath,Folder))
                                        {
                                                  SetDlgItemText(hDlg, IDC_EDIT, Folder);
                                        }
                    return TRUE;
                    //도움말 버튼
                    case IDC_HELP:
                    DialogBox( g_hInstance, MAKEINTRESOURCE(IDD_Help), HWND_DESKTOP, HelpProc );
                              return TRUE;

                    //만든이 버튼

                    case IDC_MAKER:
                              DialogBox( g_hInstance, MAKEINTRESOURCE(IDD_MAKER), HWND_DESKTOP,
MakerProc );
                    return TRUE;

                    case IDCANCEL:
                              EndDialog(hDlg,0);
                              return TRUE;
                    }
                    return FALSE;
          }
          return FALSE;
}



INT_PTR CALLBACK OkProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam)
{
          HDC hdc;
PAINTSTRUCT ps;
HANDLE hFile;
DWORD dwRead;
RECT rt;

switch(Message)
{
case WM_PAINT :

                    hdc=BeginPaint(hDlg, &ps);
                    GetClientRect(hDlg,&rt);
                              // 텍스트 바탕 출력을 투명으로
                    ::SetBkMode( hdc, TRANSPARENT );
                    DrawText(hdc,buf,-1,&rt,DT_WORDBREAK);
                    EndPaint(hDlg, &ps);
                    break
```

```
case WM_CLOSE :
{
EndDialog( hDlg, 0 );
}
}
return FALSE;
}



INT_PTR CALLBACK UnlockEditProc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)
{
		int i=0;
	switch(Message){
	//USB 인식부분
	case WM_DEVICECHANGE:
		switch (LOWORD(wParam))
		{
			case DBT_DEVICEARRIVAL:
				wcscpy(Tempname,CTxtname);
				wcscat(Tempname, (LPCTSTR)Txtname);
				while(1){
				wcscpy(copyTxtname,fTxtname[i]);
				wcscat(copyTxtname,(LPCTSTR)Txtname);
					if(PathFileExists(copyTxtname))
				{
				//MessageBox(hWnd, L"파일있음",TEXT("확인"),MB_OK);// 파일있음
				R_hFile = CreateFile(copyTxtname, GENERIC_READ, 0, NULL,
					OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
				ReadFile(R_hFile, Rstring, 20, &RRead, NULL);
				f_hFile = CreateFile(Tempname, GENERIC_READ, 0, NULL,
					OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
				ReadFile(f_hFile, fstring, 20, &RRead, NULL);
				//MessageBox(hWnd, Rstring, L"확인", MB_OK);
				//MessageBox(hWnd, fstring, L"확인", MB_OK);
		if(wcscmp(fstring, Rstring)==0)
		{
			//MessageBox(hWnd, L"일치",TEXT("알림"),MB_OK);
			g_bUnlock = true
			CloseHandle(R_hFile);
			CloseHandle(f_hFile);
			EndDialog( hWnd, 0 );
			break
		}
		else
		{
			CloseHandle(R_hFile);
			CloseHandle(f_hFile);
			MessageBox(hWnd, L"암호불일치", L"오류", MB_OK);
			break
		}
```

```
                    break
                    }
                    else //파일없음
                    {
                              i++;
                              if(i==6){i=0; MessageBox(hWnd, L"파일없음", L"오류", MB_OK); break}
                                        }
                                        }
                                        break

                    case DBT_DEVICEREMOVECOMPLETE:
                               g_bUnlock = false
                              break
                    default:
                              break
                    }
                    return TRUE;

return 0;
          }
          return FALSE;
}
INT_PTR CALLBACK HelpProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam)
{
HDC hdc;
PAINTSTRUCT ps;
HANDLE hFile;
DWORD dwRead;
RECT rt;

switch(Message)
{
case WM_INITDIALOG :
                         hFile = CreateFile(L"help.txt", GENERIC_READ, 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
                    if(hFile != INVALID_HANDLE_VALUE)
                    {
                                        ZeroMemory( buf, sizeof(buf) );
                                        ::ReadFile(hFile, buf, sizeof(buf), &dwRead, NULL);
                                        CloseHandle(hFile);
                    }
                    return 0;

case WM_PAINT :
                    hdc=BeginPaint(hDlg, &ps);
                    GetClientRect(hDlg,&rt);
                              // 텍스트 바탕 출력을 투명으로.
                    ::SetBkMode( hdc, TRANSPARENT );
                    DrawText(hdc,buf,-1,&rt,DT_WORDBREAK);
                    EndPaint(hDlg, &ps);
                    break
```

```
case WM_CLOSE :
{
EndDialog( hDlg, 0 );
}
}
return FALSE;
}
INT_PTR CALLBACK MakerProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam)
{
HDC hdc;
PAINTSTRUCT ps;
HANDLE hFile;
DWORD dwRead;
RECT rt;


switch(Message)
{
case WM_INITDIALOG :

                        hFile = CreateFile(L"Maker.txt", GENERIC_READ, 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
                        if(hFile != INVALID_HANDLE_VALUE)
                        {
                                        ZeroMemory( buf, sizeof(buf) );
                                        ::ReadFile(hFile, buf, sizeof(buf), &dwRead, NULL);
                                        CloseHandle(hFile);
                        }
                        return 0;


case WM_PAINT :
                        hdc=BeginPaint(hDlg, &ps);
                        GetClientRect(hDlg,&rt);
                                // 텍스트 바탕 출력을 투명으로.
                        ::SetBkMode( hdc, TRANSPARENT );
                        DrawText(hdc,buf,-1,&rt,DT_WORDBREAK);
                        EndPaint(hDlg, &ps);
                        break



case WM_CLOSE :
{
EndDialog( hDlg, 0 );
}


}
return FALSE;
}
```

```
//경로설정
#include <shlobj.h>
int CALLBACK BrowseCallbackProc(HWND hwnd,UINT uMsg,LPARAM lParam,LPARAM lpData)
{
	switch (uMsg) {
	case BFFM_INITIALIZED:
		if (lpData != NULL) {
			SendMessage(hwnd,BFFM_SETSELECTION,TRUE,(LPARAM)lpData);
		}
		break
	}
	return 0;
}

BOOL BrowseFolder(HWND hParent, LPCTSTR szTitle, LPCTSTR StartPath, TCHAR *szFolder)
{
	LPMALLOC pMalloc;
	LPITEMIDLIST pidl;
	BROWSEINFO bi;

	bi.hwndOwner = hParent;
	bi.pidlRoot = NULL;
	bi.pszDisplayName = NULL;
	bi.lpszTitle = szTitle ;
	bi.ulFlags = 0;
	bi.lpfn = BrowseCallbackProc::;
	bi.lParam = (LPARAM)StartPath;

	pidl = SHBrowseForFolder(&bi);

	if (pidl == NULL) {
		return FALSE;
	}

	SHGetPathFromIDList(pidl, szFolder);

	if (SHGetMalloc(&pMalloc) != NOERROR) {
		return FALSE;
	}

	pMalloc->Free(pidl);
	pMalloc->Release();
	return TRUE;
}

//문자열"[-" "-]" 제거
char Remove(char* in_hasGenre)
{
	char *bp, *tp;

bp = in_hasGenre; tp = in_hasGenre + strlen(in_hasGenre);
```

```
// 전방으로"[-"제거처리
    while ( *bp != '\0' && (*bp == '[' || *bp == '-'|| (unsigned char)(*bp) <= 0x20  ) ) bp++;
// 후방으로"-]" 제거처리
    while ( tp >= in_hasGenre && (*tp == ']' || *tp == '-'|| (unsigned char)(*tp) <= 0x20) ) tp--;
*(tp+1) = '\0'
        tp = in_hasGenre;
        while ( *bp != '\0' )
{
if ( *bp == ' '||*bp == '-'  ) { bp++; continue  }
*tp = *bp;
tp++; bp++;
}
*tp = '\0'
        return *in_hasGenre;
}
//폴더복사
// XCopy(Dir1, Dir2)형태로사용한다. 1>2
BOOL XCopy(LPCTSTR Src, LPCTSTR Dest)
{
        HANDLE hSrch;
        WIN32_FIND_DATA wfd;
        BOOL bResult=TRUE;
        TCHAR WildCard[MAX_PATH];
        TCHAR SrcFile[MAX_PATH];
        TCHAR DestFile[MAX_PATH];
        wsprintf(WildCard,L"%s\\*.*",Src);
         CreateDirectory(Dest,NULL);
        hSrch=FindFirstFile(WildCard,&wfd);
        if (hSrch == INVALID_HANDLE_VALUE)
                return FALSE;
        while (bResult) {
                wsprintf(SrcFile,L"%s\\%s",Src,wfd.cFileName);
                wsprintf(DestFile,L"%s\\%s",Dest,wfd.cFileName);
// 서브디렉토리가 발견되면 서브디렉토리를 복사한다.
                if (wfd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
                {
                        if (lstrcmp(wfd.cFileName,L".") && lstrcmp(wfd.cFileName,L".."))
                        {
                                XCopy(SrcFile, DestFile);
                        }
                }
                else
                {
                        CopyFile(SrcFile,DestFile,FALSE);
                }
                bResult=FindNextFile(hSrch,&wfd);
        }

        FindClose(hSrch);
        return TRUE;
}
```

```
BOOL FileEncryption(LPCTSTR Src, LPCTSTR Dest,HWND hwnd)


//, LPCTSTR Dest) // Foldercp FolderCP2가 들어와야 함
{
	HANDLE		hSrch;
	HANDLE		hFile;
	BOOL bResult=TRUE;

	WIN32_FIND_DATA wfd;

//	BOOL		bResult;
	TCHAR		WildCard[MAX_PATH];
	TCHAR		findFirstFileName[MAX_PATH];
	TCHAR		findSecondFileName[MAX_PATH];
	DWORD		result;
	DWORD		tlen;
	BYTE		PASSWD[100]="TEST"
	TCHAR		buf[16]={0,};
	int			poin = 0;
	int			point;

	wsprintf(WildCard,L"%s\\*.*",Src);
	hSrch=FindFirstFile(WildCard,&wfd);

	MessageBox(hwnd, WildCard, L"WildCard", MB_OK);

	if (hSrch == INVALID_HANDLE_VALUE)
		return FALSE;

	while (bResult) {
		wsprintf(findFirstFileName,L"%s\\%s",Src,wfd.cFileName);
		wsprintf(findSecondFileName,L"%s\\%s",Dest,wfd.cFileName);

		if (wfd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
		{
			if (lstrcmp(wfd.cFileName,L".") && lstrcmp(wfd.cFileName,L".."))
			{
				XCopy(findFirstFileName, findSecondFileName);
				hFile = CreateFile(findSecondFileName,

GENERIC_WRITE|GENERIC_READ, FILE_SHARE_WRITE|FILE_SHARE_READ, 0, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, 0);

		ReadFile(hFile, buf, sizeof(buf) ,&result, NULL);
		SetFilePointer(hFile, 0, NULL, FILE_BEGIN); // 파일포인터를 파일의 처음으로 이동

		tlen=lstrlen(buf);

		CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0);
```

```
//CSP(Crystographic Service Provider) 핸들얻기

                    CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);// 해쉬만들기
                    CryptHashData(hHash, (BYTE*)PASSWD,sizeof(PASSWD) ,0);// 해쉬값계산
                    CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128
                    CryptEncrypt(hKey, 0, TRUE, 0, (BYTE *)buf, &dwDataLen, sizeof(buf));//암호화
                    CryptDestroyKey(hKey);
                    CryptDestroyHash(hHash);
                    CryptReleaseContext(hProv, 0);
                    WriteFile(hFile, buf, sizeof(tlen) ,&result, NULL);

                    //for(int i=0;i<8;i++) buf[i]=0;
                    CloseHandle(hFile); //파일닫기
                    hFile = CreateFile(L"ok.txt", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                              CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                                        WriteFile(hFile, buf, sizeof(tlen), &result, NULL);
                    for(int i=0;i<16;i++) buf[i]=0;
                                CloseHandle(hFile);
                                }
                    }
                    else
                    {
                                sprintf( (char*)MY_PASSWD, "TestPassword" );
                                CopyFile(findFirstFileName,findSecondFileName,FALSE);

                    CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0);
//CSP(Crystographic Service Provider) 핸들얻기

                    CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);// 해쉬만들기
                    CryptHashData(hHash, (BYTE*)MY_PASSWD,sizeof(MY_PASSWD) ,0);// 해쉬값계산
                    CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128

                    FileEncryption( findFirstFileName, findSecondFileName, hKey );

                    CryptDestroyKey(hKey);
                    CryptDestroyHash(hHash);
                    CryptReleaseContext(hProv, 0);

                    //CloseHandle(hFile);

                    hFile = CreateFile(L"F:\ok.txt", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                                        CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                                WriteFile(hFile, buf, sizeof(tlen), &result, NULL);
                                for(int i=0;i<16;i++) buf[i]=0;
                                CloseHandle(hFile);
                    //}
          }
                    bResult=FindNextFile(hSrch,&wfd);
          }
```

```
//암호화부분
//CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0);
//CSP(Crystographic Service Provider) 핸들얻기

//		CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);// 해쉬만들기
//		CryptHashData(hHash, (BYTE*)PASSWD,sizeof(PASSWD) ,0);//(DWORD)strlen(PASSWD), 0);

// 해쉬값계산
//		CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128
//		CryptEncrypt(hKey, 0, TRUE, 0, (BYTE *)buf, &dwDataLen, sizeof(buf));//암호화
//		CryptDecrypt(hKey, 0, TRUE, 0, pbData, &dwDataLen);//복호화
//		CryptDestroyKey(hKey);
//		CryptDestroyHash(hHash); // 해쉬없애기
//		CryptReleaseContext(hProv, 0); // CSP 핸들풀어주기

		FindClose(hSrch);
		return TRUE;
}


BOOL FileDecryption(LPCTSTR Src, LPCTSTR Dest,HWND hwnd)
//, LPCTSTR Dest) // Foldercp FolderCP2가 들어와야 함
{
		HANDLE		hSrch;
		HANDLE		hFile;
		BOOL bResult=TRUE;
		WIN32_FIND_DATA wfd;

//		BOOL		bResult;
		TCHAR		WildCard[MAX_PATH];
		TCHAR		findFirstFileName[MAX_PATH];
		TCHAR		findSecondFileName[MAX_PATH];
		DWORD		result;
		DWORD		tlen;
		BYTE		PASSWD[100]="TEST“

		char		buf[16]={0,};
		TCHAR		dbuf[16]={0,};
		int			poin = 0;
		int			point;

		TCHAR		name[MAX_PATH];

		wsprintf(WildCard,L"%s\\*.*",Dest);
		hSrch=FindFirstFile(WildCard,&wfd);

		MessageBox(hwnd, WildCard, L"WildCard", MB_OK);
```

```
	if (hSrch == INVALID_HANDLE_VALUE)
		return FALSE;
	while (bResult) {
		wsprintf(findFirstFileName,L"%s\\%s",Src,wfd.cFileName);
		wsprintf(findSecondFileName,L"%s\\%s",Dest,wfd.cFileName);
		if (wfd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
		{
			if (lstrcmp(wfd.cFileName,L".") && lstrcmp(wfd.cFileName,L".."))
			{
				hFile = CreateFile(findSecondFileName, GENERIC_WRITE|GENERIC_READ, FILE_SHARE_WRITE|FILE_SHARE_READ, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
				ReadFile(hFile, buf, sizeof(buf) ,&result, NULL);

				SetFilePointer(hFile, 0, NULL, FILE_BEGIN);  // 파일포인터를 파일의 처음으로 이동

				WriteFile(hFile, buf, sizeof(tlen) ,&result, NULL);
				//for(int i=0;i<8;i++) buf[i]=0;
				CloseHandle(hFile); //파일닫기
			hFile = CreateFile(L"F:\okk.txt", GENERIC_READ | GENERIC_WRITE, 0, NULL,
				CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
			WriteFile(hFile, buf, sizeof(tlen), &result, NULL);



		for(int i=0;i<16;i++) buf[i]=0;
		CloseHandle(hFile)  }
		}
		else
		{
			sprintf( (char*)MY_PASSWD, "TestPassword" );
//for(int i=0; i<4; i++)
//{
//point = i*4+poin;
//hFile = CreateFile(findSecondFileName, GENERIC_WRITE|GENERIC_READ,
FILE_SHARE_WRITE|FILE_SHARE_READ, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
//ZeroMemory( buf, sizeof(buf) );


//SetFilePointer(hFile, point, NULL, FILE_BEGIN);
//파일포인터를 파일의 처음으로 이동
//ReadFile(hFile, buf, 16 ,&result, NULL);

//WideCharToMultiByte(CP_ACP, NULL, buf, -1, cbuf, 16, NULL, FALSE);
		CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0);
//CSP(Crystographic Service Provider) 핸들얻기
		CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash); // 해쉬만들기
		CryptHashData(hHash, (BYTE*)MY_PASSWD,sizeof(MY_PASSWD) ,0); // 해쉬값계산
		CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128
```

```
                    FileEncryption(findFirstFileName, findSecondFileName, hKey );

                    CryptDestroyKey(hKey);
                    CryptDestroyHash(hHash);
                    CryptReleaseContext(hProv, 0);
                    /*try
                    {
                              char szHex[33];
                              //One block testing
                              CRijndael oRijndael;
                    oRijndael.MakeKey("abcdefghabcdefgh", "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 16, 16);
                    char szDataOut[17] = "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0";

                    oRijndael.DecryptBlock(buf, szDataOut);
                    //CharStr2HexStr((unsigned char*)cbuf, szHex, 16);

                    MultiByteToWideChar(CP_ACP, 0, szDataOut, -1, dbuf, strlen(szDataOut));
                    //MultiByteToWideChar(CP_UTF8, NULL, szDataOut, -1, buf, 16);
                    SetFilePointer(hFile, point, NULL, FILE_BEGIN);
                    WriteFile(hFile, dbuf, sizeof(tlen) ,&result, NULL);
                    }
                    catch(exception& roException)
                              {
                                        cout << roException.what() << endl;
                              }
                                        CloseHandle(hFile);*/
                    hFile = CreateFile(L"F:\okk.txt", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                                        CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                    WriteFile(hFile, buf, sizeof(tlen), &result, NULL);
                              for(int i=0;i<16;i++) buf[i]=0;
                                        CloseHandle(hFile);
                    //}
                    }
                    bResult=FindNextFile(hSrch,&wfd);
          }
                    FindClose(hSrch);
          return TRUE;
}

// CryptAcquireContext(&hProv, NULL, MS_ENHANCED_PROV, PROV_RSA_FULL, 0);
//CSP(Crystographic Service Provider) 핸들얻기
//CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);// 해쉬만들기
//CryptHashData(hHash, (BYTE*)PASSWD,sizeof(PASSWD) ,0);// 해쉬값계산
//CryptDeriveKey(hProv, CALG_RC4, hHash, KeyLen, &hKey); // 키만들기CALG_AES_128
//CryptImportKey( hProv, (BYTE *)encKey, length, rsaKey, CRYPT_EXPORTABLE, &aesKey );
// CryptImportKey( hProv, (BYTE *)PASSWD, sizeof(PASSWD), NULL, 0, &hKey );
//CryptGenKey( hProv, CALG_AES_128,hHash,&hKey);
// CryptEncrypt(hKey, 0, TRUE, 0, (BYTE *)buf, &dwDataLen, sizeof(buf));//암호화
// CryptDestroyKey(hKey);
//CryptDestroyHash(hHash);
// CryptReleaseContext(hProv, 0);
```

Ⅲ. 결　　론

Ⅲ-1. 기대효과

본 프로그램의 화면 잠금 기능을 적절히 사용할 경우 인가받지 않은 타인이 고의성을 갖고 악의적인 목적으로 PC를 사용하고자 접근을 시도하여도 프로그램을 완전히 종료하기 전에는 닫히지 않는 구조이기 때문에 작성중인 문서를 비롯한 PC 내의 모든 개인정보를 보호할 수 있습니다.

단 PC의 Boot 메뉴나 사용자 패스워드가 설정되어 있지 않은 경우에는 재부팅 후 여전히 위협에 노출되어 있기 때문에 기본적인 패스워드 설정이 선행되어야 합니다.

특정 폴더를 지정하여 해당 폴더 안의 파일들을 암호화 한 후 백업하는 기능의 경우 USB는 휴대성과 경량이라는 장점이 있지만 분실 후의 자료에 대한 보안은 전혀 유지되지 않으므로 분실의 우려와 분실 시 내부의 정보가 전혀 보호되지 않는다는 위험이 항상 존재하기 때문에 보안이 요구되는 자료가 저장되어 있을 경우 특히 적절히 사용할 수 있다고 판단되어 기능을 탑재하였습니다.

Ⅲ-2. 프로그램의 향후 발전, 개선 방향

1. 현재(2015. 5. 13) 화면 잠금 기능에 대하여 Window이외의 다른 운영체제에서는 동작이 확인되지 않습니다. 이는 윈도우의 C++ 기반으로 코딩되었기 때문이며 추후 기회가 생긴다면 vim이나 Shell 코드 등을 이용하여 Linux 커널에서도 동작이 가능하도록 제작을 고려하고 있습니다.

2. 현재 백업 기능의 암호화 옵션은 백업할 폴더 내의 *.txt 파일로만 한정되어 있습니다. 한컴 Office의 *.hwp나 Microsoft Office의 *.doc 파일은 문서의 Read 방식이 일반 *.txt 파일과 다소 상이한 부분이 존재하여 인식하지 못하고 있습니다. 이 또한 기회가 주어진다면 더욱 다양한 확장자를 지닌 파일에 대하여 암호화가 가능하도록 함이 목표입니다.

3. 프로그램 실행 화면의 레이아웃 디자인 부분은 시제품(v 1.0.0) 으로 제작되어 프로그램 자체의 IDD 파일을 사용하였습니다. Free Permission UI와 Button, Menu 등을 사용하여 가독성을 높이는 등 Visual적인 측면을 고려하여 재 디자인 할 예정입니다.

Ⅳ. 참고자료

Ⅳ-1. 참고 알고리즘, 블로그 및 사건, 사고 사례

Windows API
https://ko.wikipedia.org/wiki/%EC%9C%88%EB%8F%84_API

DLL
https://ko.wikipedia.org/wiki/%EB%8F%99%EC%A0%81_%EB%A7%81%ED%81%AC_%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC

SK Communications 정보 유출 사건
https://mirror.enha.kr/wiki/SK%EC%BB%B4%EC%A6%88%20%EA%B0%9C%EC%9D%B8%EC%A0%95%EB%B3%B4%20%EC%9C%A0%EC%B6%9C%20%EC%82%AC%EA%B1%B4

카드 3사 개인정보 유출 사건
http://ko.wikipedia.org/wiki/2014%EB%85%84_%EB%8C%80%ED%95%9C%EB%AF%BC%EA%B5%AD_%EA%B0%9C%EC%9D%B8%EC%A0%95%EB%B3%B4_%EB%8C%80%EB%9F%89%EC%9C%A0%EC%B6%9C_%EC%82%AC%EA%B1%B4

AES 알고리즘
http://ko.wikipedia.org/wiki/%EA%B3%A0%EA%B8%89_%EC%95%94%ED%98%B8%ED%99%94_%ED%91%9C%EC%A4%80

참고 자료(블로그)
http://www.mimul.com/pebble/default/2009/02/27/1235731380000.html
http://lyb1495.tistory.com/25

Hash 함수
http://ko.wikipedia.org/wiki/%ED%95%B4%EC%8B%9C_%ED%95%A8%EC%88%98

Screen Lock
http://en.wikipedia.org/wiki/Lock_screen

Ⅴ. 별 첨

Ⅴ-1. 발표 PPT

USB를 이용한 System Lock

2015. 6.

지도교수 : 양 정 모 교수님

Team I.U (Intelligent USB)

목 차

❖ 조원 소개 및 역할

❖ 주제 선정

❖ 제작 일정 및 제작 과정

❖ 개발 환경

❖ 프로그램 동작 방식

❖ 소스코드 및 실행 화면

❖ 결론

조원 소개 및 역할

조 원	담 당 역 할
조 장 조 상 일	프로젝트 총괄
조 원 조 한 슬	화면 잠금, PW인증, 파일 백업 코딩
조 원 유 승 우	프로그램 구상, 파일 암, 복호화 구현

주제 선정

주제(프로젝트 요약)

USB를 이용한 시스템 잠금 및 파일암호화

- Window + L 명령과는 다른 자체적인 화면 잠금 기능 구현
- 사용자가 지정하는 특정 파일(폴더) 암호화
- 짧은 시간 동안 자리를 비울 때나 중요한 내용이 수시로 변경되는 파일(폴더)을 간편하고 안전하게 보호하며 백업

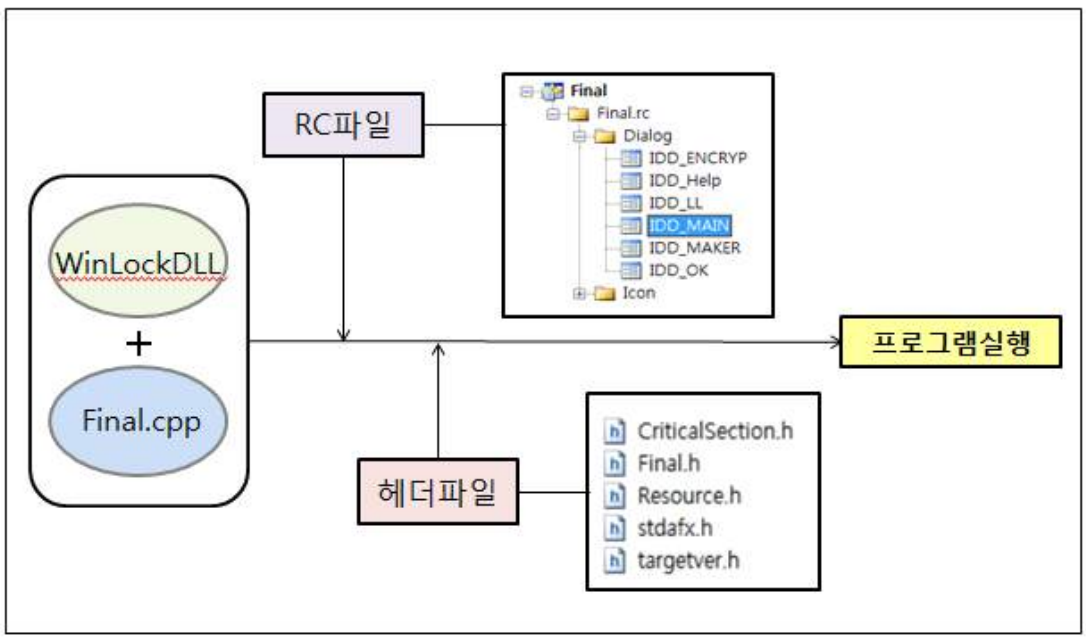
주제 선정 사유

- 스마트폰 등 휴대용 스마트 기기의 보급이 급증되었으나 많은 사용자는 보안의식 부족으로 화면 잠금 등의 기능을 사용하지 않는 경향이 있음
- 최근 휴대용 기기 분실에 따른 개인정보 유출 피해가 늘어나면서 보안의 중요성이 부각됨에 따라 1차적 도난피해 방지 프로그램을 개발

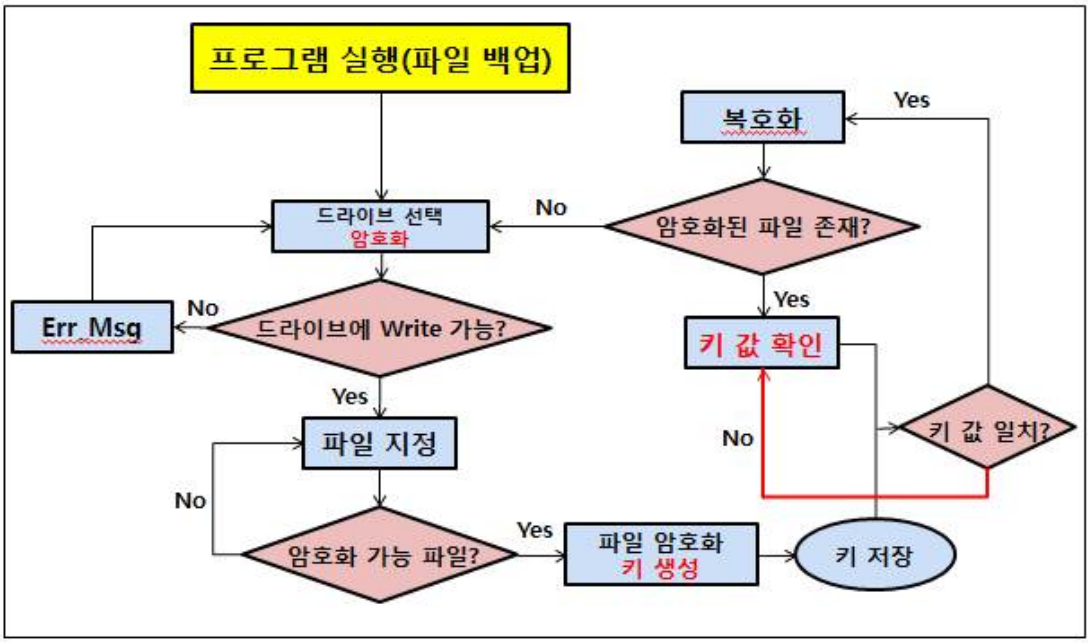
제작 일정 및 제작 과정

	2014.				2015.				
	9	10	11	12	1	2	3	4	5
구상 및 설계	■	■							
자료수집 및 개인 연구		■	■	■					
프로그램 개발				■	■	■	■	■	
USB 연결 및 PW 인증 구현				■	■	■			
AES 암호 적용						■	■	■	
프로그램 테스트 디자인, 보고서 작성								■	■

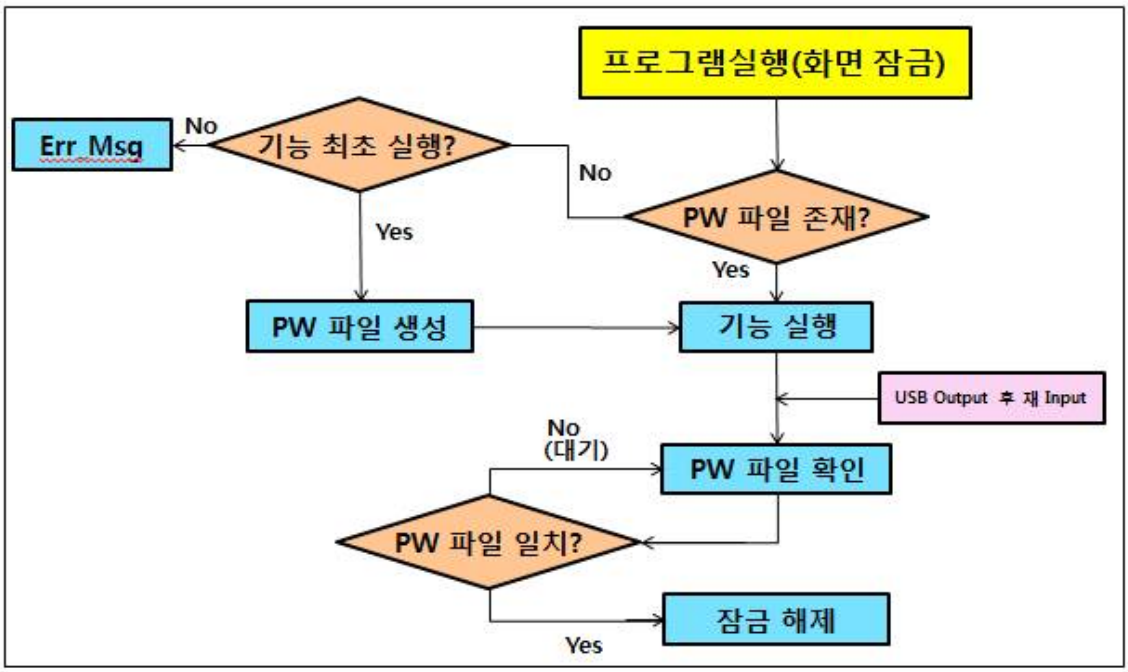
프로그램 동작 방식 (구성도)



프로그램 동작 방식 (파일 백업 Flow Chart)



프로그램 동작 방식 (화면 잠금 Flow Chart)



소스 코드

▪ 전체적인 소스코드

```
#define     WIN32_LEAN_AND_MEAN
#include "stdafx.h"
#include "resource.h"
#include "../../WinLockDLL/WinLockDLL/winlockdll.h"
#include "Shlwapi.h"
#include <dbt.h>
#include<windows.h>//파일복사
#include <shlwapi.h>//LockKey.txt 파일검사
#include <WinCrypt.h>//파일암호화

#pragma comment(lib, "Shlwapi.lib")
#define     PROGRAM_MANAGER "Program Manager"   // Program m
#define     TASKBAR         "Shell_TrayWnd"     // Taskbar c


#define    MY_PASS       "NALNARI"           // 패스워드 //ㅍ
#define    KeyLen        0x0080 * 0x10000    // 128-bit //파

HHOOK       hHook;          // Mouse hook
HINSTANCE   hInst;          // Instance handle
HINSTANCE    g_hInstance;
HWND hWnd ;

//파일생성 및 쓰기
static  HANDLE  hFile;
static DWORD    result;

static int      GStr;
//파일읽기
static  HANDLE R_hFile;
static  HANDLE f_hFile;
static DWORD  RRead;
static DWORD  FRead;
```

```
INT_PTR CALLBACK UnlockEditProc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)
{
      int i=0;
    switch(Message){
          //USB인식부분
        case WM_DEVICECHANGE:
              switch (LOWORD(wParam))
              {

                  case DBT_DEVICEARRIVAL:
                      wcscpy(Tempname,CTxtname);
                      wcscat(Tempname, (LPCTSTR)Txtname);
                      while(1){
                      wcscpy(copyTxtname,fTxtname[i]);
                      wcscat(copyTxtname,(LPCTSTR)Txtname);

                      if(PathFileExists(copyTxtname))
                      {

                          R_hFile = CreateFile(copyTxtname, GENERIC_READ, 0, NULL, OF
                          ReadFile(R_hFile, Rstring, 20, &RRead, NULL);
                          f_hFile = CreateFile(Tempname, GENERIC_READ, 0, NULL, OPEN_
                          ReadFile(f_hFile, fstring, 20, &RRead, NULL);

                          if(wcscmp(fstring, Rstring)==0)
                          {

                              g_bUnlock = true;
                              CloseHandle(R_hFile);
                              CloseHandle(f_hFile);
                              EndDialog( hWnd, 0 );
```

8

소스 코드

▪ AES 알고리즘 구성

```
int     i=0,j=0;
HWND hEdit;
INT_PTR CALLBACK MainDlgProc(HWND hDlg, UINT Message, WPARAM wParam, LPARAM lParam){
    //RECT  r;

TCHAR usblist[128];//암호키생성
static TCHAR StartPath[MAX_PATH];
TCHAR Folder[MAX_PATH];
TCHAR File[MAX_PATH]={TEXT("*.")};//리스트박스
char filetest[MAX_PATH];//(([--]제거할때 )참변환


    switch(Message){
        case WM_CREATE:
            HWND hCombo;
            hCombo=GetDlgItem(hDlg, IDOK);
            SetFocus(hCombo);
        return TRUE;

        //USB 선택 리스트박스
        case WM_INITDIALOG:
            DlgDirList(hDlg, File, IDC_LIST2, 0, (DDL_DRIVES));

            return TRUE;


        //USB인식부분
        case WM_DEVICECHANGE:
              switch (LOWORD(wParam))
              {
                  case DBT_DEVICEARRIVAL:
                      DlgDirList(hDlg, File, IDC_LIST2, 0, (DDL_DRIVES));
                      break;

                  case DBT_DEVICEREMOVECOMPLETE:
                      DlgDirList(hDlg, File, IDC_LIST2, 0, (DDL_DRIVES));
                      break;
                  default:
                      break;
              }
              return TRUE;
```

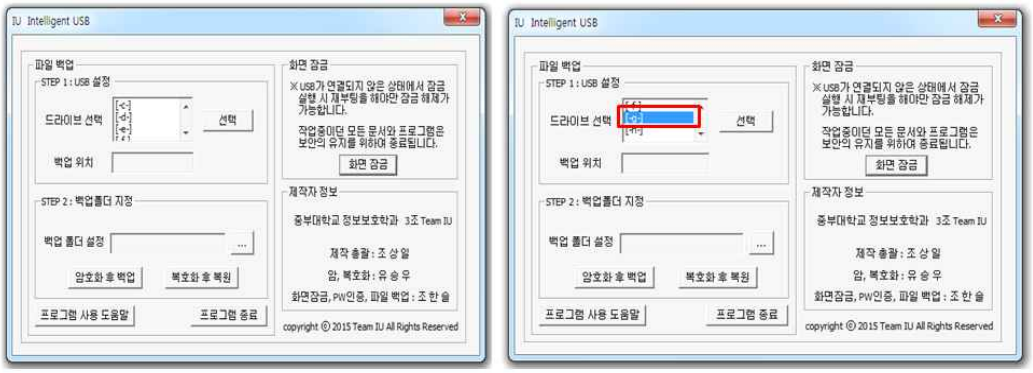
```
int getSBoxValue(int num)
{
    int sbox[256] =  {
    //0     1    2      3     4    5     6     7      8    9     A      B    C     D     E     F
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, //0
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, //1
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, //2
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, //3
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, //4
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, //5
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, //6
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, //7
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73, //8
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, //9
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, //A
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, //B
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, //C
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, //D
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, //E
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 }; //F
    return sbox[num];
}
int Rcon[255] = {
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
    0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
    0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
    0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
    0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
    0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
    0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
    0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
    0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
    0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
    0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
    0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
    0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
    0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
    0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb
};
```

9

실행 화면

메인화면

USB 연결 시



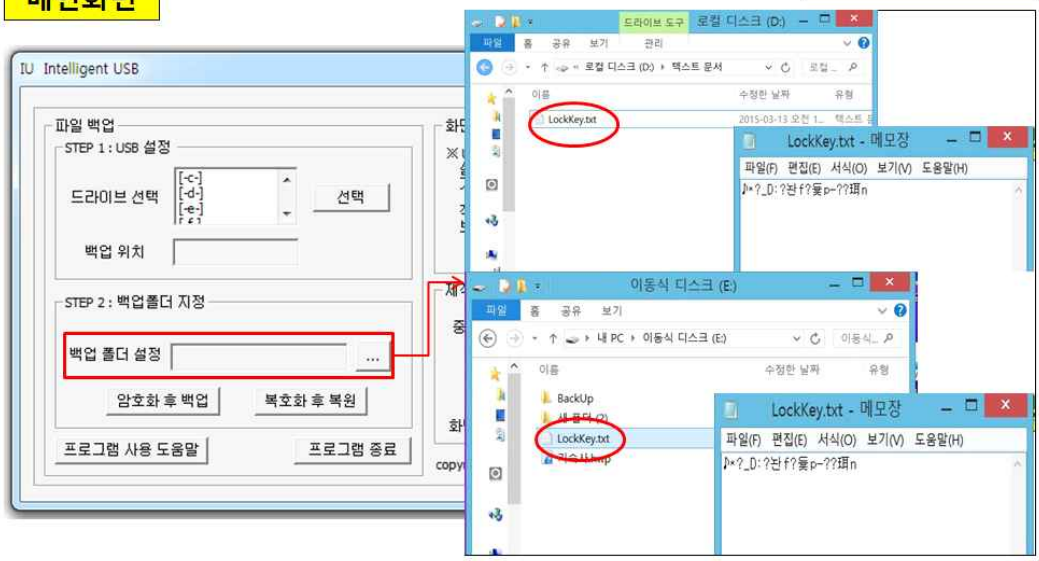
I.U 프로그램이 실행중인 USB(G:₩)가 Connect되어 백업이 가능해진 상태

10

실행 화면

메인화면

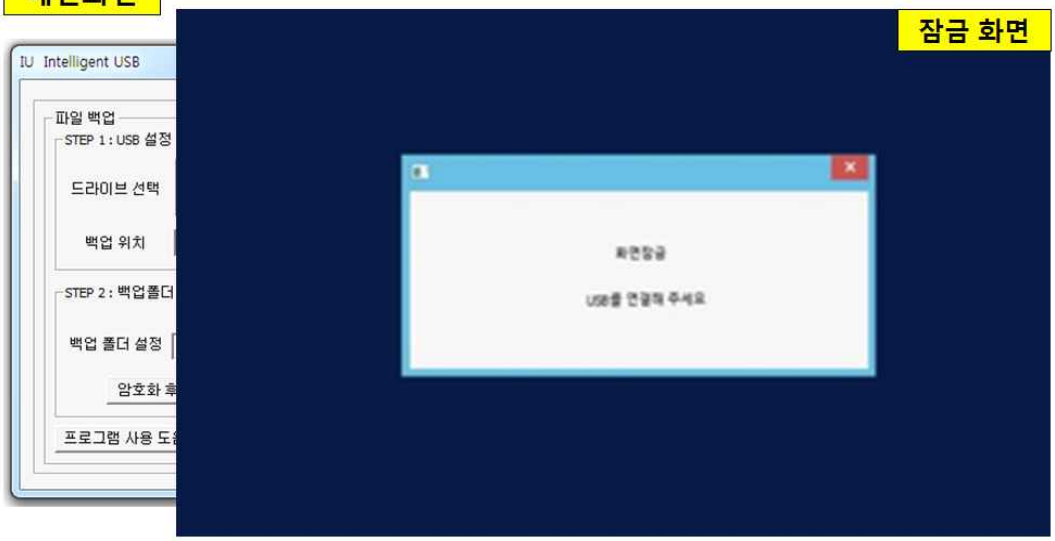
암호 키 생성



11

실행 화면

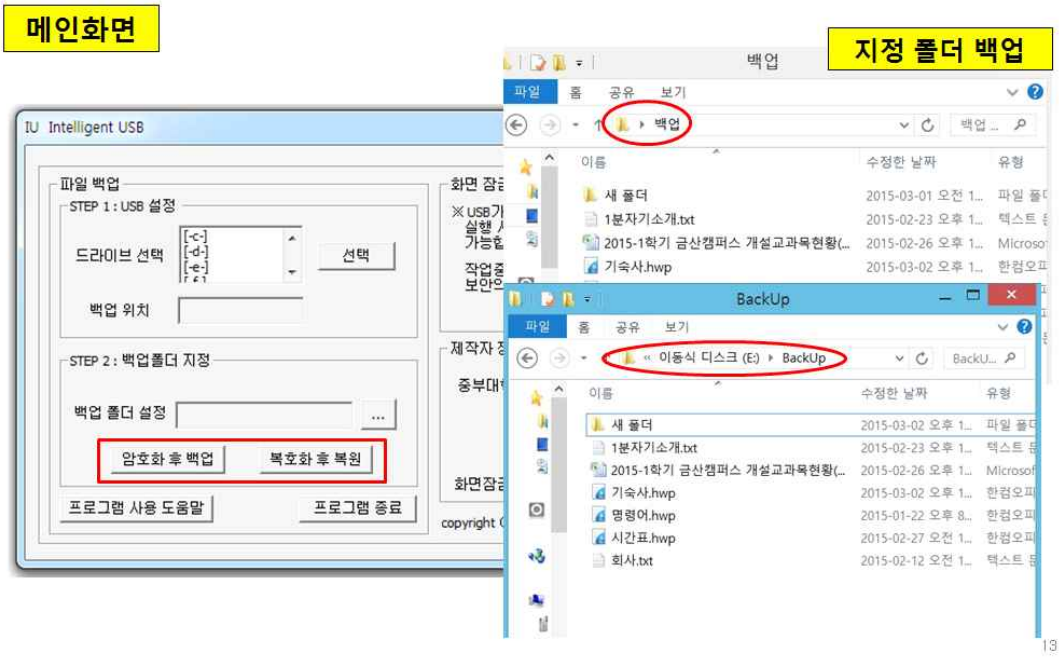
메인화면



12

실행 화면

메인화면



13

결론

기대효과

- 인가 받지 않은 타인이 고의성을 갖고 악의적인 목적으로 PC를 사용하고자 접근을 시도하여도 작성중인 문서를 비롯한 PC 내의 모든 개인정보를 보호할 수 있습니다.

- USB 안의 파일을 암호화하여 만에 하나 분실 사고가 발생해도 정보 유출 등의 보안 사고를 최소화할 수 있습니다.

향후 발전 및 개선 방향

- 윈도우의 C++ 기반으로 코딩 되었기 때문에 기타 운영체제에서는 화면 잠금 기능이 동작하지 않는 단점

- 백업의 암호화 기능은 *.txt 파일로만 제한되어있는 단점

Q&A?

감사합니다