이동형 저장매체를 위한 보안솔루션의 설계 및 구현

이선호*, 이임영*
*순천향대학교 컴퓨터학과
e-mail:[sunho431, imylee]@sch.ac.kr

Desing and Implement of A Security Solution For USB Memory

Sunho Lee*, Imyoung Lee*
*Dept of Computer Science, Soonchunhyang University

요 약

현재 많은 사용자를 확보 하고 있는 USB메모리의 분실 및 도난을 통한 개인 정보 유출 사건이 증가하고 있다. 이를 방지하기 위해서 보안USB를 이용하는 사용자가 증가하고 있지만 사용의 불편함 및 보안 취약점으로 인한 문제점이 발견되고 있다. 본 논문에서는, 기존의 보안USB의 보안영역 제공 방식을 분석하고, FAT32(File Allocation Table) 파일시스템의 구조를 이용해 보다 높은 수준의 보안을 제공하며, 보다 사용하기 쉽고 높은 확장성을 제공하는 보안영역을 설계 및 구현 하는데 목표를 두고 있다.

1. 서론

USB(Universal Serial Bus)메모리는 USB플래시 드라이브 혹은 USB 디스크 등으로 불리기도 하며 USB 포트에 꽂아 쓰는 플래시 메모리를 이용한 이동형 저장 장치를 말한다. USB메모리는 크기가 매우 작아 휴대하기도 매우 간편하며, 용량은 점차 늘어나고 가격은 저렴해지고 있어 이미 많은 사용자를 확보 하고 있다. 전자신문사와 온라인 리서치 전문업체인 엠브레인이 10대 이상 남·여 2000명을 대상으로 조사한 'USB 메모리 사용현황 조사'자료에 따르면 네티즌 응답자(2000명)의 66.4%가 USB 메모리를 가지고 있는 것으로 나타났다. 또한, USB 메모리 비보유자(672명)를 대상으로 향후 구매 의향을 묻는 질문에는 82.2%가 구매할 의향이 있다고 응답해 USB 메모리의 사용자 는 더욱 증가할 것으로 분석되고 있다.

(2007년 12월 KISA USB 메모리 보안기술 분석 참조) '일본 네트워크 보안협회'에서 개인 정보의 유출 원인을 분석한 결과 웜이나 바이러스로 인하여 개인 정보 유출이 되기보다는 도난이나 분실이 원인인 경우가 많은 것을 알 수 있다. (2007년 1월 보안 뉴스 참조) USB메모리 사용이 보편화됨에 따라 USB메모리의 분실 혹은 도난이 개인 정보 유출의 주된 원인이 되고 있는 것이다. USB메모리를 통한 개인 정보 유출을 막기 위해서 시중에 많은 보안 USB 제품이 출시되고 있지만 이 또한 불편한 사용 방법 및 보안 취약점등을 가지고 있다. 본 논문은 이와 같은 문제를 해결하기 위해서 FAT32 파일시스템의 구조를 이용하여 보다 강력한 보안을 제공하며, 보다 사용하기 쉽고 높은 확장성을 제공하는 보안 영역 제공 방식을 제안하다.

2. 관련연구

보안USB의 보안제공 방식은 크게 2가지로 분류 할 수 있다. 하드웨어를 사용하는 방식과 소프트웨어를 사용하는 방식이다. 하드웨어를 사용하는 방식의 경우 USB포트와 메모리 사이에 보안을 제공하기 위한 칩이 존재하여 사용자 인증 과정을 거친 뒤 메모리의 전원을 공급하는 등의 방식을 사용하여 강력한 보안을 제공하지만 구현하기가 힘들고 추가 비용이 발생하는 등의 문제점, 회로의 조작을 통하여 메모리의 내용을 읽어 들일 수 있는 취약점이 존재하고 있다. 소프트웨어를 사용하는 방식의 경우 USB메모리의 제조사 웹페이지에서 보안 프로그램을 다운 받아 USB메모리를 사용할 컴퓨터에 설치하여 사용하는 방식으로 가장 많은 보안USB가 사용하는 방식이다. 위 방식을 사용하는 몇몇 재품은 사용자 인증 과정의 스니핑을 통하여 사용자 인증을 위한 비밀번호나 그 힌트가 평문으로 노출되는 취약점, 일반영역에 보안영역의 드라이브 이미지 파일을 숨겨 놓아 일반 파티션의 포맷을 통하여 보안 영역도 같이 삭제되는 취약점, 일반영역 포맷 뒤 데이터 복구 프로그램을 이용하여 비밀번호 없이 보안 영역의 데이터에 접근할 수 있는 취약점, 사용자 인증 뒤 로그오프를

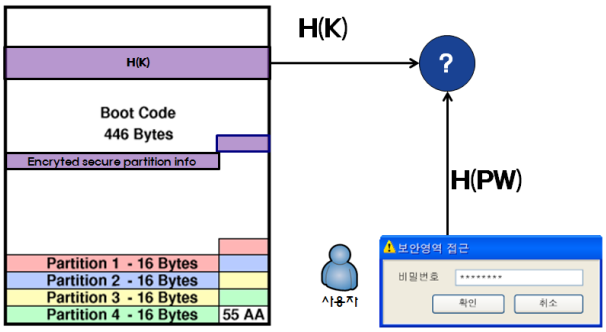
하지 않고 메모리를 제거 뒤 다시 컴퓨터에 USB메모리를 인식 하였을 때 바로 보안영역이 인식되는 취약점, 보안영역의 크기를 사용자 임의로 설정하지 못하는 불편함, 존재하고 있다.

이와 같은 여러 가지 단점들을 보완하기 위해서 본 논문에서는 다음과 같이 비밀번호 평문 노출을 막기 위한 비밀번호의 해시값을 이용한 사용자 인증 기술, 일반영역과 보안영역의 물리적 분리를 위한 강재 다중 파티셔닝 및 보안영역 파티션 정보 암호화 기술, 안전한 보안영역 재설정을 위한 태그기반 보안영역 설정 기술, USB메모리의 비정상적인 제거 시에도 보안영역을 노출하지 않는 파티션 스와핑 기술 등을 사용한 USB 보안솔루션을 제안한다.

3. 설계

3.1 사용자 인증

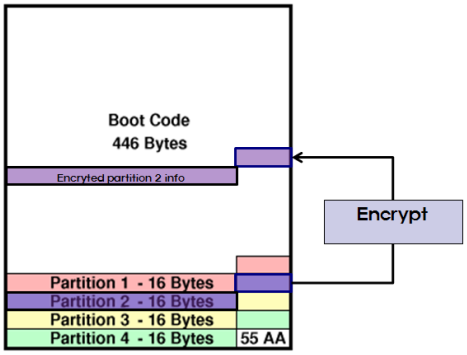
사용자 인증을 위해서 사용자가 설정한 비밀번호와 입력한 비밀번호를 비교하는 과정이 필요하다. 기존의 USB 메모리 보안프로그램 대부분이 비교를 위한 비밀번호를 평문으로 저장하는 방식을 사용해 보안에 많은 위험이 있었다. 이를 해결하기 위해서 그림 1과 같이 사용자가 입력한 비밀번호의 해시값을 저장하였다가 사용자가 입력한 암호의 해시값과 비교하여 사용자 인증을 하는 방식을 사용한다.



(그림 1) 해시값을 이용한 사용자 인증 구조

3.2 보안영역 제공

물리적으로 분리된 보안영역을 구현하기 위해 본 논문에선 다중 파티션을 이용한다. 하지만 USB메모리의 경우 다중 파티션을 지원하지 않을 뿐 더러 강재로 설정한다고 해도 파티션 테이블1번에 설정된 파티션만 인식하게 된다. 이와 같은 USB메모리의 성질과 FAT파일시스템의 구조를 이용하여 파티션을 강제 할당하여 임의로 접근이 불가능한 강력한 보안 파티션을 생성 할 수 있게 된다. 또한 이 보안파티션 정보를 노출하지 않기 위해 그림 2와 같이 파티션 테이블 1번에 일반파티션을 저장, 파티션 테이블 2번에 저장될 보안파티션의 정보를 사용자가 설정한 암호로 암호화 저장하여 파티션 테이블이 아닌 MBR(Master Boot Record)의 사용하지 않는 공간에 저장한다.



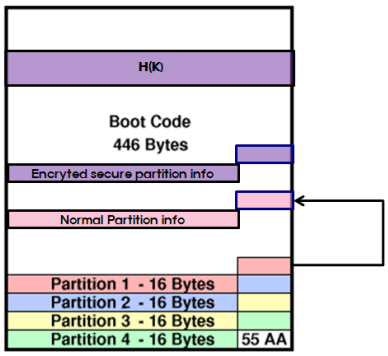
(그림 2) 보안 파티션 정보 암호화 저장

3.3 보안영역 보호

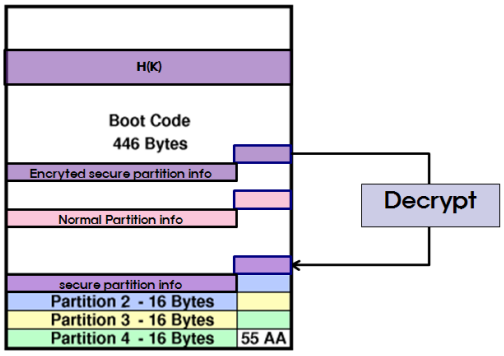
그림한번 보안 파티션이 할당된 뒤 다시 제 3자로부터 보안 파티션이 재 할당 되면 이미 설정된 보안 파티션이 훼손될 가능성이 있을 수 있다. 위와 같은 상황을 막기 위해서 보안영역 파티셔닝 시 특정 태그를 삽입하여 특정 태그가 존재할 경우 정당한 사용자임을 증명하기 위해 로그인을 요청하도록 설계가 하였다. MBR임을 증명하는 55AA 매직 코드와 흡사한 기능을 하는 코드를 삽입하는 것이다. 본 논문에선 0번 섹터에 위치하고 있는 MBR의 첫 번째 블록부터 4byte의 길이로 0x42425607 라는 임의로 설정한 코드를 삽입하여 보안 파티션의 존재 여부를 나타내도록 하며, 차후 해당 코드가 존재할 경우 사용자 인증을 거쳐야만 보안영역 재설정이 가능하도록 한다.

3.4 파티션 스왑

사용자 인증을 통하여 정당한 사용자임을 증명한 뒤 보안 파티션 마운트를 시도하게 되면 보안 파티션의 정보를 복호화 하여 파티션 테이블 1번에 파티션 정보를 넣게 된다. 보안 파티션 정보를 넣기 전 파티션 테이블 1번에 저장된 일반파티션의 정보가 손상되는 것을 막기 위해 그림 3과 같이 특정위치에 일반파티션의 정보를 저장하게 된다. 그 뒤 그림 4와 같이 암호화된 보안 파티션의 정보를 사용자가 입력한 비밀번호로 복호화 하여 파티션 테이블 1번에 기록한다.

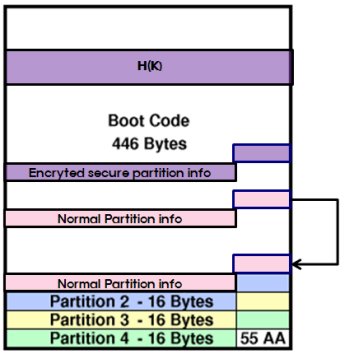


(그림 3) 일반영역 파티션 정보 백업



(그림 4) 보안영역 파티션 정보 복호화

위 과정까지만 진행하게 된다면 보안 파티션 인식 중 USB 메모리를 제거 후 제 3의 컴퓨터에 인식 시킬 경우 바로 보안 파티션으로 인식될 위험이 있다. 이것을 방지하기 위해서 그림 5와 같이 보안 파티션이 인식된 직후 바로 특정위치에 저장해둔 일반 파티션의 정보를 파티션 테이블 1번에 덮어 쓰게 된다. 이는 윈도우가 인식중인 저장매체의 MBR 조작에 바로 반응하지 않고 인식을 요청하거나 다음 인식 시 적용 되는 특성을 이용한 기술이다.



(그림 5) 일반영역 파티션 덮어쓰기

파티션 스왑 과정에서 변경된 파티션의 정보를 PC에 업데이트하기 위해서 그림 6과 같은 API 코드 DeviceIoControl을 이용한다.

```
BOOL WINAPI DeviceIoControl(
  _in        HANDLE hDevice,
  _in        DWORD dwIoControlCode,
  _in_opt    LPVOID lpInBuffer,
  _in        DWORD nInBufferSize,
  _out_opt   LPVOID lpOutBuffer,
  _in        DWORD nOutBufferSize,
  _out_opt   LPDWORD lpBytesReturned,
  _inout_opt LPOVERLAPPED lpOverlapped);
```

(그림 6) DeviceIoControl 함수

dwIoControlCode에 FSCTL_DISMOUNT_VOLUME를 할당 하여 현제 마운트된 파티션을 디스마운트 시킨 뒤, 다시 IOCTL_DISK_UPDATE_PROPERTIES를 할당하여 바뀐 파티션의 정보를 업데이트 하게 된다.

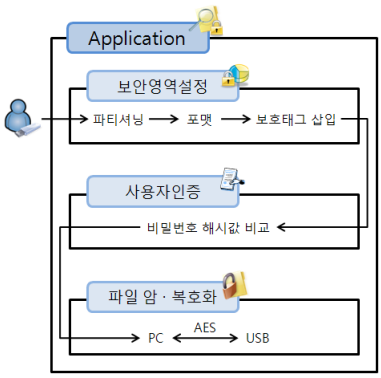
3.5 파일 암호화

위 기능들을 사용하여 안전한 접근제어가 가능하지만 더욱 강화된 보안을 제공하기 위해서 파일 암호화 기능이 추가 제공되어야 한다. 본 논문에서는 AES 암호화 알고리즘을 이용하여 2중의 보안을 제공한다.

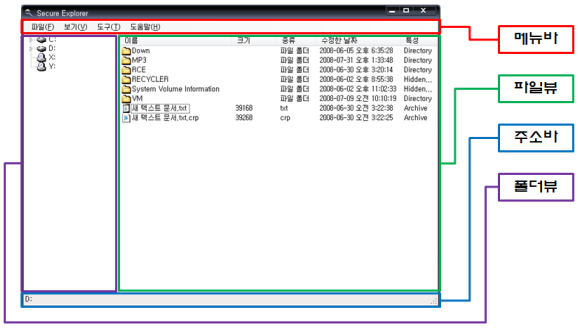
4. 구현

본 논문에서 제안한 사용자 인증, 보안영역 제공, 보안영역 보호, 파티션 스왑, 파일 암호화 기술을 사용해서 어플리케이션을 구현하였다. 또한 사용자에게 친숙한 UI(User Interface) 제공을 위해서 윈도우즈 운영체제에서 사용하는 파일 탐색기와 매우 흡사한 디자인으로 어플리케이션을 구현하였다.

그림 7은 사용자가 USB메모리를 PC에 연결한 뒤 본 논문에서 구현한 어플리케이션을 통하여 보안서비스를 제공 받는 순서도를 나타낸 것이다.



(그림 7) 어플리케이션 구조



(그림 8) 어플리케이션 인터페이스

<표 1> 메뉴바 세부내용

상위메뉴	하위메뉴	세부메뉴	기능
파일	끝내기		프로그램 종료
보기	큰아이콘		파일뷰 큰 아이콘으로 보기
	작은아이콘		파일뷰 작은 아이콘으로 보기
	간단히		파일뷰 간단히 보기
	자세히		파일뷰 자세히 보기
	새로고침		파일뷰, 폴더뷰 새로고침
도구	포맷		보안영역 설정 및 포맷
	로그인		설정된 보안영역 접근을 위한 사용자 인증
	마운트	일반영역	일반영역마운트
		보안영역	보안영역마운트
	암호화		파일 암호화
	복호화		파일 복호화

5. 결론

본 논문에서는 USB메모리의 보다 안전하고 편리한 사용 환경을 위해서 사용자에게 익숙한 인터페이스, 비밀번호 해시 값을 이용한 안전한 사용자 인증 방식, FAT32 파일 시스템의 구조를 이용해 보안 파티션 생성 및 파티션 정보를 암호화 하여 안전한 보안 영역 제공 방식, 어떠한 상황에서도 보안 파티션의 정보를 노출하지 않기 위해 윈도우즈 운영체제의 드라이브 인식 방법의 특성을 이용한 일반 파티션과 보안 파티션의 스왑 기능을 설계 및 구현 하여 사용자에게 더욱 안전하고 친숙하며 확장성 높은 보안 영역을 제공하였다.

본 논문은 보안 기능이 강화되고 구현비용이 적게 드는 방식을 제안하여 보안USB를 생산하는 업체에서 유용하게 사용될 것이다.

참고문헌

[1] 최용락, 소우영, 이재광, 이임영, “컴퓨터 통신보안“, 그린출판사, 2006

[2] “Hardware White Paper - FAT : General Overview of On-Disk Format”, Microsoft Corporation, 1999.

[3] 정한재, 최윤성, 전웅렬, 양비, 김승주, 원동호 “보안 USB 플래시 드라이브의 취약점 분석과 CC v3.1 기반의 보호프로파일 개발”, 정보보호학회논문지 제17권 제6호, 2007. 12, pp. 99 ~ 119 (21pages)

[4] 이혜원, 박창욱, 이근기, 김권엽, 이상진, “포렌식 관점에서의 보안 USB 현황분석”, 2008년도 한국방송공학회 동계 학술대회, 2008. 2, pp. 63 ~ 65 (3pages)

[5] 고찬, 박연, "RSSS 방식에 의한 USB Driver의 보안기능 강화", 2005

[6] http://www.usb.org

[7] http://msdn.microsoft.com

[8] http://www.cryptosystem.net/aes