ETRI

2013
Electronics and
Telecommunications
Trends

사이버 보안 기술 특집

안전 저장장치 기술 동향

A Technical Trend of Secure Storages

박종연 (J.Y. Park) 모바일보안연구실 연구원
임재덕 (J.D. Lim) 모바일보안연구실 선임연구원
김정녀 (J.N. Kim) 모바일보안연구실 실장

* 본 연구는 방송통신위원회의 ETRI연구개발지원사업의 연구결과로 수행되었음(KCA-11921-05001).

PC 및 스마트기기의 사용이 일반화됨에 따라, 최근 개인의 민감한 정보가 외부로 유출되는 사고가 빈번히 발생하고 있다. 그럼에도 불구하고, 일반적인 사용환경에서는 개인의 데이터가 평문 파일 형태로 관리되고 있는 실정이며, 이러한 상황에서는 공격자가 수월하게 공인인증서 및 개인정보 유출이 가능하다. 본고에서는 데이터를 보다 안전하게 저장 · 관리하기 위한 안전 저장장치에 대한 개념 및 기술 동향에 대하여 살펴본다. 특히, 현재까지 사용되고 있는 소프트웨어와 하드웨어 기반의 안전 저장장치 전반에 걸친 기술 동향 및 각 모듈의 특징을 분석하고 향후 연구 방향에 대하여 논한다.

Ⅰ. 서론

최근 스마트기기 사용이 보편화되고, 누구나 손안에 작은 컴퓨터를 들고 다니는 시대가 되었다. 이동이 자유로울 뿐만 아니라 어디에서든지 네트워크와 연결되고 GPS와 WiFi를 통한 정밀한 위치인식이 가능해져 스마트기기들은 무한히 많은 응용 프로그램과 연계되었다. 이러한 스마트기기에는 금융거래를 위한 인증서, 위치기반 정보, 사진 등의 개인정보를 저장할 수 있어, 사용자에게 편의성을 제공해 주는 반면, 저장된 개인정보의 유출 위험이 늘 존재한다.

민감한 데이터가 저장되어 있고 항상 네트워크에 접속되어 있으며, 여러 출처의 애플리케이션을 설치하여 이용하는 스마트기기의 특성상 다양한 해킹 방법이 존재해 왔으며, 안드로이드 OS 기반의 악성코드의 경우 최근 1년 사이에 30배 이상 증가된 것으로 발표되었다[1]. 최근에는 공격자가 간단한 애플리케이션 제작을 통하여 스마트폰에 저장되어 있는 인증서 정보를 탈취할 수 있음이 알려졌다[2]. 이와 같이 개인의 중요 정보가 노출될 수 있는 위험성이 존재함에 따라 공격에 대한 적절한 대응기법에 대한 연구가 요구되고 있다. 그 중 하나의 방법은 데이터를 안전하게 저장하고자 하는 안전 저장장치(secure storage)이다.

안전 저장장치는 기존에 데스크톱 PC의 window와 리눅스 기반의 OS에서 구동하도록 개발된 것이 일반적이다. 데이터 저장장치에 대한 위협은, 1차적으로 안전하게 저장하여 해킹으로부터 데이터를 안전하게 보호하고, 2차적으로는 물리적으로 데이터를 탈취하더라도 저장된 데이터를 인증되지 않은 사용자로 하여금 접근을 통제하는 것을 의미한다. 더 나아가서 지워진 데이터를 복구해 내는 포렌식 기술에 대항하여 데이터를 안전하게 삭제하는 방법 또한 안전 저장장치에서 고려해야 할 기술이다.

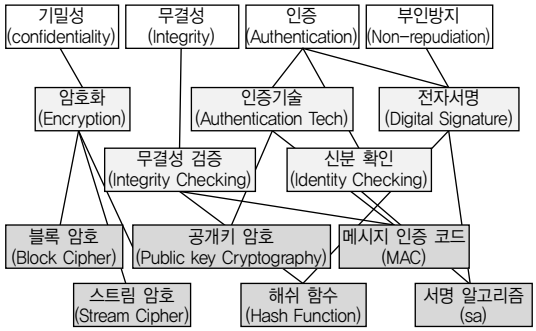
본고는 현재까지 알려진 안전한 저장장치 솔루션의 특징을 살펴보고, 다양한 측면으로 개발 결과물을 비교하여 연구자와 개발자들에게 향후 연구에 필요한 요구사항을 도출해 낼 수 있도록 기술한 동향 보고서이다. 본고의 Ⅱ장에서는 보안 사전지식을 설명하고, Ⅲ장에서는 소프트웨어, 하드웨어 기반의 안전 저장장치에 대하여 기술할 것이며, Ⅳ장에서 본고를 결론 지을 것이다.

Ⅱ. 보안 사전지식

1. 암호 기술

안전한 저장장치에 사용되는 필수적인 기술은 암호 기술이다[3]. 암호(cryptography)는 기본적으로 읽을 수 있는 메시지를 읽을 수 없는 메시지로 변환하는 것과 다시 복원하는 기술을 의미하며, 단순히 암/복호화하는 작업뿐만 아니라, 인증, 서명 등의 다양한 기능을 총칭한다((그림 1) 참조).

일반적으로 사용되는 암호 기술은 블록 암호(block cipher), 스트림 암호(stream cipher), 공개키 암호(public key cryptography), 해시 함수(Hash Function), 메시지 인증 코드(message authentication code), 서명 알고리즘(signature algorithm) 등으로 이루어진다. 각 암호 기술은 암호화된 메시지의 노출을 방지하는 기밀성(confidentiality), 메시지의 수정을 방지하는 무결성(integrity), 사용자의 신분(identity)를 확인하는 절차를 포함한 포괄적 인증



(그림 1) 암호 기술 종류의 개념도

(authentication), 사용자의 행위에 대한 거부를 방지하는 부인방지(non-repudiation) 기술을 이용하기 위함이다. 이러한 암호 기술은 안전 저장장치에서도 동일하게 사용되며, 일반적으로 기밀성과 무결성, 인증 기술을 적용·이용한다.

암호화 및 인증 기능은 안전저장 장치에서 필수적으로 사용되며, 안전 저장장치의 솔루션의 대부분은 암호 기술에 대한 의존도가 높다. 따라서 암호 알고리즘의 강도가 낮으면 저장 장치의 보호 매커니즘이 아무리 잘 설계되어 있다고 하더라도 안전한 저장장치라고 보기 어렵다.

2. 안전 저장장치의 필수적 요구사항

안전 저장장치는 안전하게 저장한다는 기본적인 개념은 물론이고, 사용편의성과 무결성 검증까지 포함한다[4]. 암호 기술 소개에서 설명한 바와 같이 안전한 저장장치에 필수적 보안기술 요구사항은 기밀성, 무결성, 그리고 인증 기술이다. 그리고 추가적으로 저장장치를 사용하는 사용자의 불편을 최소화 하기 위한 이용성(availability)이 중요한 요소이다.

가. 기밀성

저장장치의 기밀성을 유지하기 위해서 고려되어야 할 사항은 다음과 같다

- 사용된 암호 알고리즘의 종류
- 암호화를 위해 사용된 키를 저장하는 방법
- 암호화를 위해 사용된 키를 추출하는 방법
- 암호화 키의 수명
- 암호화를 위해 사용된 암호모드(modes of operation)의 종류
- 암호화하여 저장하는 데이터의 영역(데이터, 파일 메타데이터, 파일 이름, 디렉토리 등)

사용자가 안전 저장장치를 이용하기 위해서는 데이터를 암호화하여 저장하는 것뿐만 아니라, 저장된 데이터를 복호화할 수 있어야 한다. 그래서 전체 구조 하에서 암호화 키를 반드시 저장하고 있어야 하는데, 이때에 사용된 키가 안전하게 저장되어 있지 않다면 암호화하여 저장하더라도 안전하다고 볼 수 없다. 따라서 키가 어떻게 추출, 저장되는지에 대한 매커니즘이 중요하다. 또한 사용된 암호 알고리즘의 종류나, 암호 모드의 종류에 따라서 저장장치의 안전도 여부가 결정되게 된다[5].

나. 무결성, 인증, 이용성

저장장치에 저장된 데이터는 공격자에 의해서 수정될 수도 있으며, 공격자의 불법적인 접근에 대하여 통제되어야 한다. 무결성 검증 기술에 대한 요구사항은 다음과 같다.

- 사용된 해시 함수의 종류
- MAC(Message Authentication Code) 값 생성을 위해 사용된 암호 알고리즘의 종류
- MAC 값 생성을 위해 사용된 키 관리 방법

단순히 해시 함수만을 이용하는 무결성 검증은 키가 들어있지 않으므로, 안전하다고 볼 수 없다. 그 이유는 데이터의 수정과 동시에 해시 값도 수정할 수 있기 때문이다. 따라서 기밀성 유지와 같이 키가 안전하게 관리되어야 함이 필수적인 요구사항이다.

- 불법적인 사용자의 접근통제 방법
- 보안정책의 수정가능 여부
- 사용자 패스워드의 길이, 수명, 수정가능의 편의성

위의 내용은 인증 및 이용성에 대한 검증사항이다. 이외에도 많은 요구사항과 검증사항이 있을 수 있다.

Ⅲ. 안전 저장장치 기술

안전 저장장치 기술에 대한 다양한 분류 방법이 존재할 수 있다. 본 장에서는 Sarah[4]의 분류 방법을 통해 안전 저장장치 기술에 대하여 설명할 것이다. 안전 저장장치는 크게 소프트웨어 기반의 기술과, 하드웨어 기반 기술로 나뉠 수 있다.

1. 소프트웨어 기반 기술

가. 암호화 프로그램

가장 널리 사용되는 소프트웨어 기반의 기술은 암호화 프로그램이다. 암호화 프로그램은 유저가 데이터 저장 프로그램을 구동시켜 저장하는 방식이다. 본 방식은 애플리케이션 단에서 수행되기 때문에 기능적인 유연성이 높다. 이러한 방법의 예로는 한글, 마이크로소프트 워드, 파워포인트 등에서 단순히 암호화하여 저장하는 방법도 포함된다. 하지만 암호화 프로그램은 단순히 사용자 패스워드 정도를 입력하여 저장하는 것이 일반적이므로, 안전한 방법이라고 볼 수 없다. 더욱이 파일 핸들링을 위한 메타데이터가 암호화되지 않으므로 데이터 자체가 암호화되어 있다고 하더라도, 파일의 존재 여부가 확인 가능하다. 소프트웨어적으로 유연하게 관리되므로, 저장 암호화 모듈, 저장 프로그램 동작 방식 결정, 보안 정책 관리 등이, 다른 방식보다 쉽게 바꾸는 것이 가능하다.

일반적으로는 공개된 오픈 소스를 이용하여 지키고 싶은 데이터를 암호화하여 저장한다. 가장 널리 사용되는 소프트웨어는 Open SSL(Secure Socket Layer)[6], OpenPGP 표준을 따르는 Gnu Privacy Guard, GunPG[7] 등이 있다. 본 방식은 SSL에 사용되는 오픈 소스(open sourse)로서 잘 구현된 암호 모듈로 알려져 있다. 〈표 1〉은 암호화 프로그램의 장·단점을 정리한 것이다.

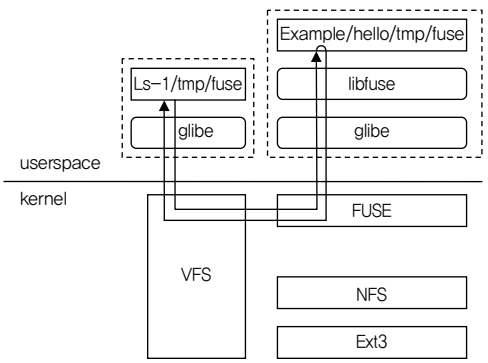
〈표 1〉 암호화 프로그램의 특징

장점	단점
- 유연한 정책 관리 - OS에 의존성 없이 독립된 프로그램으로 사용	- 상대적으로 안전성이 높지 않다 - VFS layer cashing을 사용할 수 없으므로 상대적으로 퍼포먼스가 떨어질 수 있음
사용 예) Open SSL, GunPG, 한글, MS 워드 등 각종 애플리케이션 암호화 저장 등	

나. 사용자 영역 파일 시스템

사용자 영역 파일 시스템은 FUSE(Filesystem in Userspace)[8]를 이용하여 안전 암호 파일 시스템을 만든 것이다. FUSE는 유닉스 커널에서의 VFS(Virtual File System)을 이용하지만, 커널을 수정하지 않고 사용하는 응용도구이다. (그림 2)는 FUSE를 이용한 파일 시스템의 흐름도이다. VFS 위에 다른 파일 시스템과 같이 FUSE가 존재하고, FUSE를 루트로 사용자 공간에서 다른 암호화 파일 시스템이 구동된다.

FUSE를 이용한 대표적인 파일 시스템은 EncFS[9]와 CryptoFS[10], MetFS[11], Lessfs[12] 등이 있다. EncFS는 대표적인 FUSE 기반의 파일 시스템으로써, 파일을 암/복호화 하는 것이 투명하게 관리된다. EncFS는 CFS(Cryptographic File System)[13]의 디자인을 대부분 따르고 있으며, CFS를 기본으로 확장된 형태를 갖추고 있다. EncFS는 데이터를 암호화하여 저장하지만, 암호화되



〈자료〉: http://fuse.surceforge.net

(그림 2) FUSE를 이용한 파일 시스템 흐름도

〈 표 2〉 사용자 영역 파일 시스템의 특징

장점	단점
– 비교적 유연한 정책 관리 – 암호화 프로그램 기반 파일 암호화와 비교하여 상대적으로 빠른 속도	– FUSE 자체의 연산 오버헤드
사용 예) EncFS, CryptoFS, MetFS, PhoneBookFS, Lessfs 등	

〈표 3〉 Stackable 파일 시스템의 특징

장점	단점
– 안전한 키 관리 – 다양한 기능 지원 및 PKI 지원 – 파일 단위 관리의 효율성	– 디렉토리 구조와 파일 메타데이터의 공개 – 별도의 레이어 구조 때문에 발생하는 연산 오버헤드
사용 예) Ncryptfs, Cryptfs, Ecryptfs 등	

어 있는 파일의 개수, 파일의 권한(permission), 파일 크기 등의 메타데이터는 암호화하지 않고 노출되어 있다. 〈표 2〉은 사용자 영역 파일 시스템의 장단점을 정리한 것이다.

다. Stackable 파일 시스템

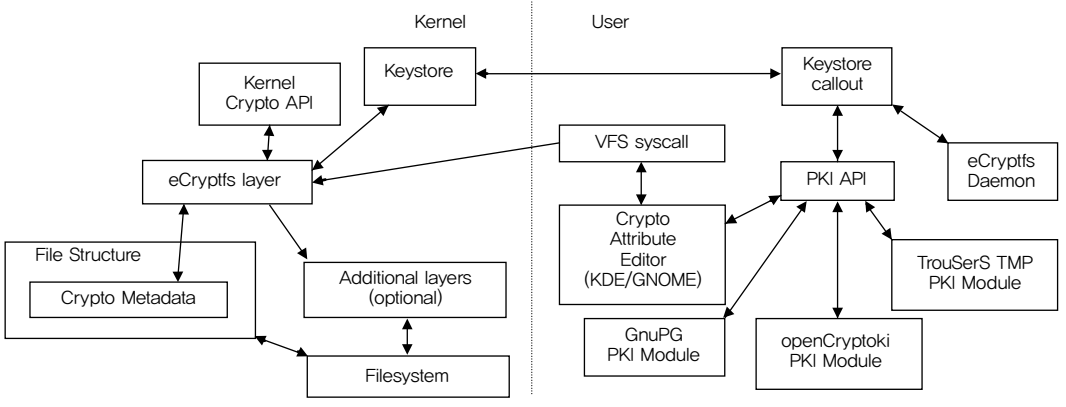
Stackable 파일 시스템은 별도의 데몬 없이, 커널 레벨에서의 시스템 콜을 인터셉트하여 이용 가능한 구조를 말한다. 대표적인 stackable 암호 파일 시스템은 Cryptfs[14], NCryptfs[15], eCrpytfs[16]등이 있다.

Cryptfs와, Ncryptfs는 FiST[17]의 애플리케이션이다. FiST는 개발자들로 하여금 high-level에서 커널 파일 시스템을 생성하여 다양한 플랫폼에서 이용 가능하게 해준다.

eCryptfs는 PAM(Pluggable Authentication Module), TPM(Trusted Platform Module), GunPG와 같이 커널 수준의 암호 파일 시스템 API에서는 거의 사용되지 않는 공개키 암호까지도 이용하여 통합 솔루션을 제공한다[16].

eCryptfs는 커널의 변화를 요구하지 않으며, 암호화 파일을 가상파일 시스템(VFS)레이어 위에 미리 준비된 디렉토리 안에 마운트하여 관리한다. (그림 3)은 eCryptfs의 암호 및 키 관리 구조를 나타낸다. 커널에서 관리되는 keystore와 사용자 영역에서의 keystore callout을 이용하여 PKI API를 통해 PKI 모듈로 키를 이용할 수 있도록 만든다. 키는 cryptographic meta data를 특별한 파일 안에 내장하는 방식을 취하며, 파일 단위로 암/복호화를 수행한다. 이러한 정책에 대한 내용 역시 별도의 파일로 관리한다. 키는 디스크에 저장되지 않으며, TPM을 이용하여 하드웨어적으로 관리한다. 하드웨어 키를 소실하면 세션 키를 유도할 수 없게 된다.

Stackable 파일 시스템은 일반적으로 디스크에 키를 저장하지 않으며, temporary 파일은 암호화된 디렉토리에 따로 저장한다. 하지만 파일 메타데이터와 디렉토리 구조는 노출되어 있다는 약점이 존재한다. 그리고 모든 VFS 레이어 위에서 기능이 간접적으로 동작하므로 다른 파일 시스템보다 더 많은 오버헤드가 발생한다는 약점이 있다. 〈표 3〉은 stackable 파일 시스템의 장단점을 정리한 것이다.



(그림 3) eCryptfs 암호 및 키 관리 구조[16]

라. 블록 기반 파일 시스템

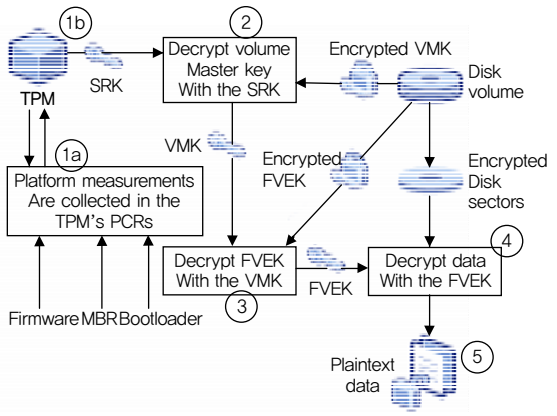
블록 기반의 파일 시스템은 소프트웨어 암호화 파일 시스템 중 가장 낮은 단계에서 동작한다. 대표적인 블록 기반 시스템은 dm-crypt[18], BestCrypt[19], the Crypto-Graphic Disk driver[20], Encrypted Volume and File System[21], Microsoft BitLocker Drive Encryption[22] 등이 있다.

dm-crypt는 리눅스 버전 2.6에서 동작하는 sub system으로써, 커널의 Crypto API를 이용한다. dm-crypt는 리눅스 버전 2.5에서 처음 소개된 Cryptloop의 확장된 버전으로 볼 수 있다. Cryptloop가 watermark 공격에 취약한 것과는 달리[23] dm-crypt는 ESSIV(Encrypted Salt-Sector Initialization Vector)를 이용하여 CBC모드에서의 약점을 보완하였다.

BitLocker[22]은 full disk encryption을 지원하며, 마이크로소프트 윈도우 7, 8등의 데스크톱 OS부터, window server 2008, window server 2008 RS, windows server 2012 등에 적용되었다. CBC 모드와 128비트 AES를 기본으로 동작하며, AES 256비트 역시 지원한다. 더욱이 CBC 모드에서의 약점을 보완하기 위하여 Elephant diffuser[24]를 추가적으로 적용하여 사용한다. BitLocker는 디바이스의 컴퓨팅 능력에 따라서 여러 가지 옵션을 제공하며, 옵션은 다음과 같다.

- BitLocker with TPM
- BitLocker with USB device
- BitLocker with TPM and PIN(Personal Identification Number)
- BitLocker with TPM and USB

본 보고서에서는 BitLocker with TPM만 간략하게 소개하기로 한다. TPM을 이용한 BitLocker는 TPM version 1.2를 필요로 한다. 이 옵션은 사용자에게 투명하게 제공



(그림 4) BitLocker에서 TPM을 이용한 파일 복호화 방법[22]

되는데, 그 이유는 추가적인 하드웨어나, 패스워드를 필요로 하지 않기 때문이다. (그림 4)은 TPM을 이용한 복호화 과정을 나타낸 것이다.

① BIOS(Basic Input Output System)가 시작되고 TPM을 초기화한다. 암호 시스템은 TPM으로부터 SRK(Storage Root Key)를 넘겨 받는다.

② 디스크로부터 암호화된 VMK(Volume Master Key)와 암호화된 FVEK를 넘겨받는다. ①에서 받은 SRK를 이용하여 암호화된 VMK를 복호화한다.

③ 복호화된 VMK를 이용하여 FVEK를 복호화한다.

④ 복호화된 FVEK를 이용하여 데이터를 최종적으로 복호화한다.

데이터는 TPM으로부터 받은 SRK를 이용하여 키를 추출하기 때문에, 실제 데이터를 복호화하는 데 사용되는 FVEK는 TPM 모듈이 없이는 추출하기 어렵다.

블록 기반의 암호화 시스템은 다른 방식보다 상대적으로 투명하게 사용자가 이용 가능하며, 전체 디스크가 모두 암호화되므로, 포렌식 공격으로부터 상대적으로 안전하다. 〈표 4〉은 블록 기반 파일 시스템의 장·단점을 정리한 것이다.

〈표 4〉 블록 기반 파일 시스템의 특징

장점	단점
– 커널 레벨에서 동작하므로 빠른 퍼포먼스 – TPM 등을 이용하여 키를 추출하고 관리하므로 안전하고 투명함. – 포렌식 공격으로부터 높은 안전성 확보	– 키와 암호 모드에 대한 유동성 부족
사용 예) dm-crypt, BestCrypt, BitLocker, EVFS 등	

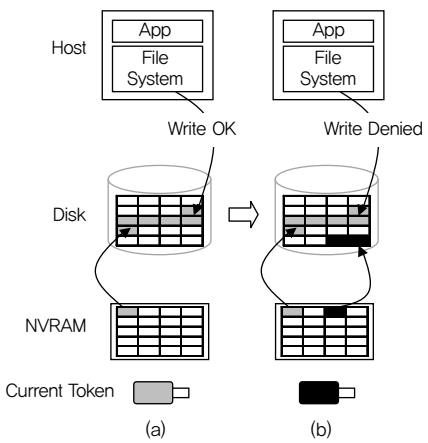
〈표 5〉 하드웨어 기반 파일 시스템의 특징

장점	단점
– 가장 빠른 퍼포먼스 – 사용이 쉽고 투명하게 사용 가능 – 높은 안전성 – 사용 및 개발을 위한 많은 비용	– 키와 암호 모드를 바꾸는 것이 전혀 불가능
사용 예) Secure Ironkey, Kingston DataTraveler Secure, RocSecure Hard Drives, SecureDisk Hardware, Seagate 등	

마. 저장장치를 이용한 Rootkit 방지 기술

Rootkit이란, 공격자가 시스템을 해킹하기 위하여 설치하는 악성 프로그램으로써, Kernel, Bootkits, Hypervisor, Hardware/Firmware 레벨까지 침투하여, 백도어, 트로이목마, 내부 사용흔적 삭제, 관리자 권한 획득 등을 제공하는 프로그램을 말한다[25].

RRD(Rootkit-Resistant Disk)[26]은 rootkit 방지 기술로써, 일반 모드와 관리자 모드로 나누어서 시스템이 관리된다. (그림 5)는 RRD를 이용한 저장장치 관리 방법을 나타낸다. (그림 5a)는 회색 USB 토큰을 이용하여 할당된 블록에 시스템 정보를 저장한 모습이다. (그림 5b)는 저장된 정보를 수정 또는 덮어 씌우기 위하여 검은색 USB 토큰을 이용하여 기록하려 하지만 해당되는 블록은 회색 토큰이 없이는 재기록이 되지 않고 보호된다.



(그림 5) RRD를 이용한 저장장치 관리 방법[26]

2. 하드웨어 기반 기술

하드웨어 기반의 안전 저장장치는 소프트웨어와 기본적인 개념은 유사하나, 저장을 위한 암호 기능 등이 하드웨어나 외부 저장 장치에 하드웨어 코딩되어 동작한다. 하드웨어 기반으로 연산하므로 소프트웨어 방식보다 훨씬 빠른 속도로 암/복호화를 수행한다. 하지만 고정되어 있는 방식을 수정할 수 없으므로, 유동성이 떨어지는 단점이 있다.

소프트웨어 기반의 방식에 비하여 상대적으로 하드웨어 기반의 방식을 사용하여 얻을 수 있는 안전성은 다음과 같다. 소프트웨어 방식에서 사용하는 패스워드를 통한 암호화 장치를 예로 들어보자. 일반적으로 복호화하기 위하여 공격자가 패스워드를 전수조사한다고 했을 때에, 이러한 전수조사를 막기 위하여 일정 이상의 횟수를 초과하는 시도하였을 때에 파일에 대한 접근 통제가 이루어진다. 하지만 이러한 경우 공격자는 memory rewind 공격이 가능해진다[27]. 즉, 공격자는 단순히 전수조사 전에 소프트웨어의 temporary 파일을 갱신함으로써, 무재한 전수조사 시도가 가능해 지도록 만든다. 하지만 하드웨어 기반의 방식은 password guessing counter를 하드웨어적으로 관리하여 안전성을 확보할 수 있다. 〈표 5〉은 하드웨어 기반 파일 시스템의 장단점을 정리한 것이다.

Ⅳ. 결론 및 향후 연구 방향

본고에서는 안전 저장장치 기술의 종류와 그 특징을 살

펴보고, 장단점을 논하였다. 일반적으로 소프트웨어 방식 중 상위 단계에서의 솔루션은 사용의 유동성이 높은 대신 속도 저하가 많이 발생하며, 하위 단계에서의 솔루션은 유동성이 떨어지는 반면, 속도가 크기 감소하지 않음을 알 수 있다. 더욱이 하드웨어 기반의 방식은 가장 유동성이 떨어지면서, 하드웨어 기반으로 구현되어 있기 때문에 소프트웨어 방식과 비교하여 퍼포먼스는 상당히 높음을 알 수 있다.

본고에서 다룬 암호화 파일 시스템 및 저장 장치를 활용한 공격 방지 기술은 모바일 디바이스가 아닌 PC를 위한 솔루션이었다. 서론에서 언급한 바와 같이 모바일 디바이스의 경우 무선 네트워크가 항상 연결되어 있고, 애플리케이션의 구동시간이 PC보다 높고, 악성 코드를 이용한 공격 가능성이 커지고 있으므로, 더욱 높은 단계의 보안 대책이 요구되고 있다. 따라서 앞으로 모바일 장비에서는 안전 저장장치가 선택이 아닌 필수적으로 요구될 것이라고 전망할 수 있다.

향후에는 기존의 방법보다 높은 안전성을 확보할 수 있는 키 관리 방법 개발, 연산 효율성이 떨어지지 않는 소프트웨어 및 하드웨어 기반의 암/복호화 방법, 사용자 단계에서도 더욱 투명하게 관리되는 솔루션 개발이 이루어 져야 할 것으로 전망된다.

약어 정리

API	Application Programming interface
BIOS	Basic Input Output System
CFS	Cryptographic File System
ESSIV	Encrypted Salt-Sector Initialization Vector
FUSE	Filesystem in Userspace
FVEK	Volume Encryption Key
MAC	Message Authentication Code
OS	Operating System
PAM	Pluggable Authentication Module
PIN	Personal Identification Number
RRD	Rootkit-Resistant Disk
SRK	Storage Root Key
SSL	Secure Socket Layer
TPM	Trusted Platform Module
VFS	Virtual File System
VMK	Volume Master Key

참고문헌

[1] 안철수연구소, “2012년 모바일 악성코드, 실체적인 위협으로 다가오다,” 2013. 1. 8. http://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?menu_dist=2&seq=20375

[2]안철수연구소, “공인인증서 탈취 악성코드 주의보,” 2013, 4, 7. http://www.ahnlab.com/company/site/pr/comPressRelease/comPressReleaseView.do?seq=1438083

[3] NIST, “Security Requirements for Cryptographic Modules, FIPS PUB 140-2,” Mar. 12th, 2002.

[4] S. M. Diesburg and An-I A. Wang, “A Survey of Confidential Data Storage and Deletion Method,” ACM Comput. Surveys, vol. 43. no. 1, Article no. 2, Nov. 2010.

[5] NIST, “Recommendation for Key Management – Part 1: General,” Special Publication 800-75, Mar. 2007.

[6] Open SSL Project. http://www.openssl.org

[7] Gnu Privacy Guard. http://www.gnupg.org

[8] FUSE. http://fuse.sourceforge.net

[9] EncFs. http://www.arg0.net/encfs

[10] CryptoFS. http://reboot78.re.funpic.de/cryptfs

[11] Metfs. https://code.google.com/p/metfs/

[12] Lessfs. http://www.lessfs.com/wordpress/

[13] M. Blaze, “A Cryptographic File System for Unix,” ACM Conf. Commun. Comput. Security, Faixfax Va, Nov. 3th-5th, 1993.

[14] E. Zadok, I. Badulescu, and Alex Shender, “Cryptfs: A Stackable Vnode Level Encryption File System,” Computer Science Department, Columbia University, Feb. 1999.

[15] C. Wright, M. Martino, and E. Zadok, “NCryptfs: A Secure and Convenient Cryptographic File System,” Proc General Track USENIX Annual Technical Conf., June 2003.

[16] M.A. Halcrow, "eCryptfs: An Enterprise-class Cryptographic Filesystem for Linux," http://ecryptfs.sourceforge.net/ecryptfs.pdf

[17] Z. Zadok and J. Nieh, "FiST: A Language for Stackable File Systems," ATEC Proc. Conf. USENIX Annual Technical Conf., USENIX Association Berkeley, CA, 55-70, 2000.

[18] M. Peters, "Encrypting Partitions Using dm-crypt and the 2.6 Series Kernel," Linux, June 8th, 2004. http://www.linux.com/artic-les/36596

[19] BestCrypt. http://www.jetico.com

[20] R. Downeswell and J. Ioannidis, "The CryptoGraphic Disk Driver," Proc. Annual USENIX Technical Conf., Berkeley, CA, 179-186. 2003.

[21] Hewlett-Packard, "Encrypted Volume and File System v1.0.02 Release Notes," 2011. http://docs.hp.com/e -n/5992-3353/5992-3353.pdf

[22] Microsoft Corporation, "Bitlocker Drive Encryption: Exeutive Overview. Technical Report" 2012. http://technet.microsoft.com/en-us/library/hh831412.aspx

[23] SecuriTeam, "Linux Cryptoloop Watermark Exploit,". http://www.securiteam.com/exploits/5UP0P1PFPM.html, May 2005.

[24] N. Fergusson, "AES-CBC + Elephant Diffuser: A Disk Encryption Algorithm for Windows Vista," Microsoft, Aug. 2006.

[25] Halflife. "A Buse of the Linux Kernel for Fun and Profit," Phrack Mag., vol. 7, no. 50, Apr. 1997.

[26] K. Butler, S. McLaughlin, and P. McDaniel, "Rootkit-Resistant Disks," CSS, Virginia, USA, Oct. 2008.

[27] Ironkey, "Benefits of Hardware-based Encryption," White paper, Feb. 2007. http://www.futureshield.com/productArticles/IronKey_white paper on the benefits on hardware based encryption.pdf