情報保護學會論文誌
第 22 卷 第 4 號, 2012. 8

스마트폰 플래시 메모리 이미지 내의 단편화된 페이지 분석 기법 및 구현*

박 정 흠,† 정 현 지, 이 상 진,‡ 손 영 동
고려대학교 정보보호대학원

Design and Implementation of Analysis Techniques for Fragmented Pages in the Flash Memory Image of Smartphones*

Jungheum Park,† Hyunji Chung, Sangjin Lee,‡ Youngdong Son
Graduate School of Information Security, Korea University

요 약

휴대폰은 개인의 생활과 가장 밀접한 디지털 기기로 디지털 포렌식 수사 시에 반드시 고려해야 할 대상이 되고 있다. 최근에는 스마트폰의 사용이 증가하고 있는데, 스마트폰은 피처폰과는 달리 일반 PC와 유사한 고성능의 운영체제(Android, iOS 등)를 사용하면서 다양한 모바일 앱(app)을 통해 사용자에게 여러 가지 기능을 제공하는 특징이 있다.

디지털 포렌식 관점에서 스마트폰의 사용 흔적을 수집하고 분석하는 것이 중요해짐에 따라서 전세계적으로 스마트폰 포렌식에 관한 연구가 활발하게 이루어지고 있다. 스마트폰 내의 데이터를 분석하기 위해서는 백업 또는 디버깅 기능을 이용하여 사용자 파일을 추출하거나, 운영체제의 루트(root) 권한을 획득하여 플래시 메모리에 대한 이미지를 수집한 후에 파일시스템(YAFFS, EXT, RFS, HFS+ 등)을 재구성하여 분석하는 방법을 사용할 수 있다. 그러나 이와 같은 방법은 정상적으로 존재하는 파일 이외에 삭제되거나 플래시 메모리 페이지의 데이터가 수정되면서 생성될 수 있는 잉여 데이터에 대한 분석에는 한계가 있다.

본 논문에서는 스마트폰으로부터 획득한 플래시 메모리 이미지 내의 단편화된 페이지(fragmented pages)를 분석하는 기법을 소개한다. 이를 통해 플래시 메모리 페이지의 스페어 영역(spare area)이 없어서 파일시스템의 재구성이 불가능한 이미지 또는 정상적인 파일시스템의 비할당 영역에 속하는 임의의 페이지들에 대한 효과적인 분석 방법을 제시한다.

ABSTRACT

A cell phone is very close to the user and therefore should be considered in digital forensic investigation. Recently, the proportion of smartphone owners is increasing dramatically. Unlike the feature phone, users can utilize various mobile application in smartphone because it has high-performance operating system (e.g., Android, iOS).

As acquisition and analysis of user data in smartphone are more important in digital forensic purposes, smartphone forensics has been studied actively. There are two way to do smartphone forensics. The first way is to extract user's data using the backup and debugging function of smartphones. The second way is to get root permission, and acquire the image of flash memory. And then, it is possible to reconstruct the filesystem, such as YAFFS, EXT, RFS, HFS+ and analyze it. However, this methods are not suitable to recovery and analyze deleted data from smartphones.

This paper introduces analysis techniques for fragmented flash memory pages in smartphones. Especially, this paper demonstrates analysis techniques on the image that reconstruction of filesystem is impossible because the spare area of flash memory pages does not exist and the pages in unallocated area of filesystem.

Keywords: Digital Forensics, Smartphone Forensics, Flash Memory, Unallocated Area, Fragmented Data

접수일(2012년 3월 28일), 수정일(2012년 5월 14일), 게재확정일(2012년 6월 24일)

* 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업의 일환으로 수행되었습니다.〔10035157, 실시간 분석을 위한 디지털 포렌식 기술 개발〕

† 주저자, junghmi@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr

I. 서 론

최근 널리 사용되고 있는 스마트폰은 기존의 피처폰과는 달리 일반 PC와 유사한 운영체제를 사용하면서 다양한 기능을 제공한다. 특히 스마트폰에서 동작하는 모바일 앱(app)은 다양한 사용자 데이터를 저장 및 관리하고 있다.

스마트폰에서는 데이터를 저장하기 위해서 플래시 메모리(flash memory)를 주로 이용한다. 플래시 메모리는 작고 가벼우면서 대용량의 데이터를 저장할 수 있기 때문에 스마트폰과 같은 임베디드 기기에서 널리 사용되고 있다. 일반적으로 읽기 속도가 빠르고 가격이 비싼 NOR 플래시 메모리에는 실행 코드를 저장하며, 상대적으로 속도는 느리지만 가격이 저렴한 NAND 플래시 메모리에 사용자의 데이터를 저장한다.

디지털 포렌식 관점에서 스마트폰 내의 사용자 흔적을 수집하고 분석하는 것이 중요해짐에 따라서 전세계적으로 스마트폰 포렌식에 관한 연구, 특히 NAND 플래시 메모리에 저장되어 있는 사용자 데이터를 분석하는 기법에 관한 연구가 활발하게 진행되고 있다. 현재 일반적으로 활용되고 있는 스마트폰 포렌식 기법은 크게 두 가지가 있다. 첫 째는 스마트폰 운영체제에서 제공하는 백업 또는 디버깅 기능을 이용하여 사용자 파일을 추출하여 분석하는 것이다. 이 방법은 운영체제에서 허용하는 범위에 한해서만 데이터를 추출할 수 있어서 모든 사용자 데이터를 분석할 수 없다는 한계점이 있다. 그리고 두 번째는 운영체제의 루트(root) 권한을 획득하여 플래시 메모리 이미지를 수집한 후에 파일시스템(YAFFS, EXT, RFS, HFS+ 등)을 분석하는 것이다. 두 번째 방법은 모든 파일을 획득할 수 있다는 장점이 있지만, 플래시 메모리 이미지를 획득하는 방법에 따라서 웨어-레벨링(ware-leveling)된 페이지(page)들을 재구성하는 작업이 필요할 수 있다. (추가적으로 JTAG(Joint Test Action Group)포트를 연결할 수 있는 경우에는 루트 권한 없이도 플래시 메모리 이미지를 수집할 수 있다.)

플래시 메모리의 특성상 페이지(page) 단위로 분할된 데이터가 메모리 전역에 걸쳐서 분산되어 저장되는데, 사용하는 파일시스템과 물리 이미지를 획득하는 방법에 따라서 다양한 분석 방법이 적용된다. 예를 들어, Android 스마트폰에서 주로 사용하는 YAFFS의 경우에는 플래시 메모리 페이지들을 재구성하기 위해서 스페어 정보가 필요하기 때문에 플래시 메모리 칩을 물리적으로 추출해서 데이터를 획득하거나, nanddump와 같은 도구(플래시 메모리 드라이버에서 제공하는 명령 사용)를 이용하여 플래시 메모리 페이지와 스페어 영역을 함께 수집하는 방법을 사용할 수 있다[1][2]. 또한 iPhone(iOS)의 경우에는 루트 권한을 획득하여 마운트된 파일시스템을 dd로 이미징하면, 특별한 재구성 절차 없이 정상적인 파일시스템을 해석할 수 있다. (물론 iOS 4.0 이후에는 데이터를 암호화하기 때문에 이를 복호화 해야 정상적인 분석이 가능하다[3].)

플래시 메모리 이미지를 획득한 후에는 파일시스템을 해석하여 내부의 데이터에 대한 분석을 수행한다. 그러나 플래시 메모리의 스페어 영역(spare area)이 없어서 페이지의 재구성이 불가능하여 파일시스템을 재구성할 수 없는 경우에는 분석에 많은 어려움이 있다. 또한 마운트된 파일시스템을 이미징하여 파일시스템의 해석이 가능한 경우에는 정상적으로 존재하는 파일에 대한 분석을 수행할 수 있지만, 비할당 영역과 같이 단편화된 페이지(파일시스템 클러스터 또는 블록)들에 대한 효율적인 분석은 이루어지지 않고 있다.

본 논문에서는 스마트폰으로부터 획득한 플래시 메모리 이미지 내의 단편화된 페이지(fragmented pages)를 분석하는 기법을 소개한다. 사용하는 파일시스템에 상관없이 플래시 메모리의 페이지 (파일시스템 클러스터 또는 블록) 레벨에서 분석을 수행할 수 있다. 특히 대부분의 스마트폰에서는 특정 파일 형식을 사용하여 사용자의 데이터를 저장하기 때문에 효율적인 분석이 가능하다. 최근 플래시 메모리를 사용하는 기기가 증가하면서 단편화된 페이지를 분석하는 연구는 매우 의미가 있으며, 이러한 연구 결과는 디지털 포렌식 조사 과정에서 유용하게 활용될 수 있을 것이다.

본 논문은 다음과 같이 구성된다. 2장에서는 스마트폰에서의 플래시 메모리에 대해 설명하고, 3장에서는 디지털 포렌식 관점에서 스마트폰을 분석하는 절차를 소개한다. 4장에서는 플래시 메모리 페이지를 분석하는 기법을 설명하고, 5장에서는 본 논문에서 소개한 기법을 이용하여 실제 이미지를 대상으로 실험을 수행한다. 그리고 6장에서 향후의 연구 계획을 설명하고, 마지막으로 7장에서 결론을 내린다.

참고로 본 논문에서 사용하는 '페이지'라는 용어는 플래시 메모리의 데이터 저장 단위이지만, 파일시스템의 비할당 영역을 대상으로 할 때는 각 파일시스템에 따라서 '클러스터' 또는 '블록'을 의미한다.

II. 스마트폰에서의 플래시 메모리

플래시 메모리는 페이지(page) 단위로 데이터를 저장하며, 동일한 페이지가 계속 사용되는 것을 방지하기 위해 웨어-레벨링을 수행한다. 즉 웨어-레벨링에 의해 페이지 크기 이상의 데이터를 저장할 때는 페이지 단위로 분할된 데이터가 메모리 전역에 분산 저장된다. 이러한 기능을 지원하기 위해서는 파일시스템 관점의 논리적인 위치와 플래시 메모리 상의 물리적인 위치를 매핑(mapping)하는 정보가 필요하며, 이 매핑 정보는 각 페이지의 스페어 영역과 메타 페이지라는 특수 용도의 페이지 등에 저장된다.

또한 플래시 메모리에서는 기존 데이터를 업데이트한다는 개념이 존재하지 않는다. 즉 데이터를 업데이트할 때는 기존의 페이지를 삭제하고, 새로운 페이지를 할당하여 데이터를 기록한다. 이 때 플래시 메모리에서는 블록(페이지의 모음) 단위로만 삭제되기 때문에 많은 경우에 수정 이전의 데이터가 그대로 남아있을 가능성이 있다. 이와 같은 플래시 메모리의 특성은 스마트폰에서도 매우 중요하게 활용될 수 있는데, 문자, 전화, 일정 등의 사용자 데이터가 업데이트됨에 따라 그 내역이 남게 되어 사용자가 특정 시점에 데이터를 삭제했더라도 상당수의 이전 데이터를 복구할 가능성이 있다.

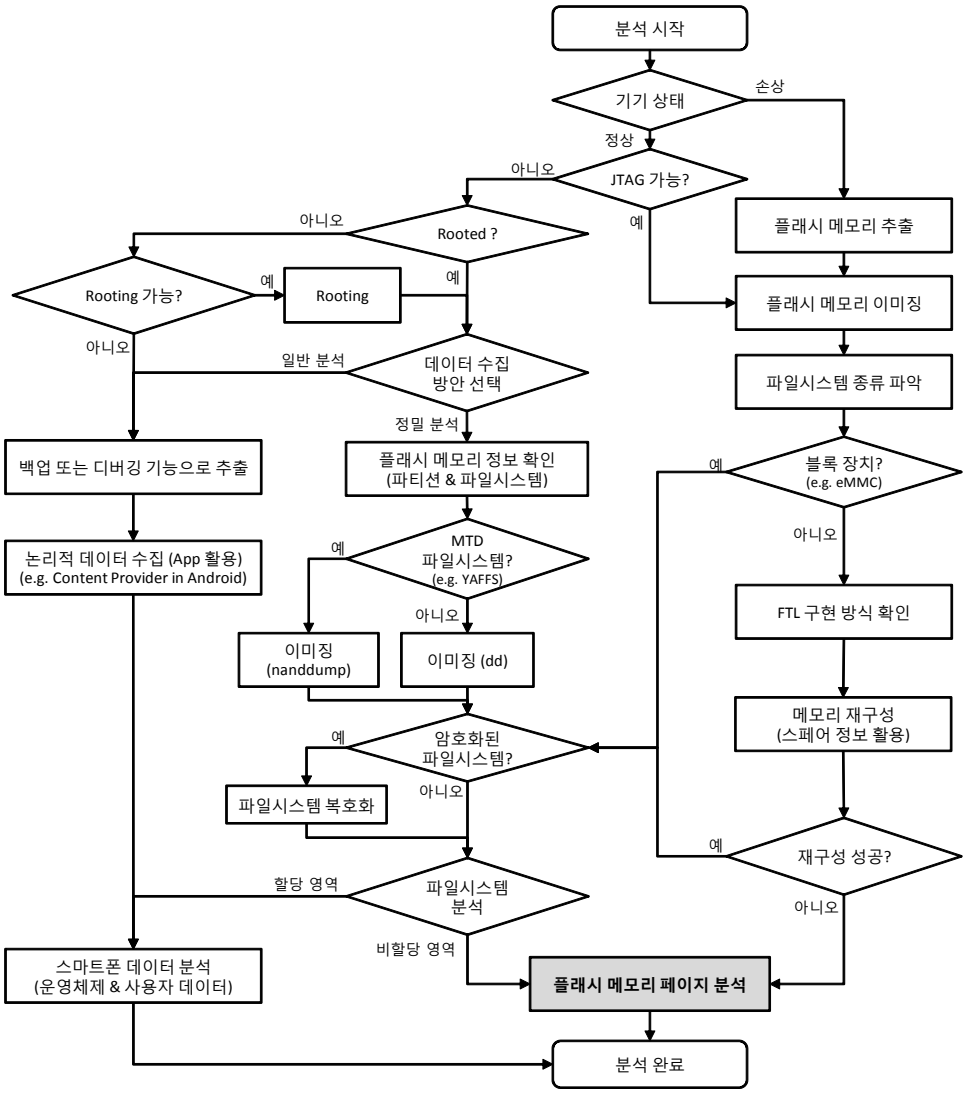
한편 스마트폰에서는 데이터를 효율적으로 관리하기 위해 일반 PC와 유사하게 파일시스템을 사용한다. 현재 널리 사용되는 스마트폰 운영체제에는 Google의 Android, Apple의 iOS, RIM의 Blackberry OS, Nokia의 Symbian 등이 있으며, 이러한 운영체제가 적용된 스마트폰에서는 YAFFS (Yet Another Flash File System), EXT (Extended File System), RFS (Robust File System), HFS (Hierarchical File System) 등의 다양한 파일시스템을 사용하고 있다. 따라서 디지털 포렌식 조사 과정에서 다양한 파일시스템에 대한 이해는 필수적이다. 특히 YAFFS와 같은 MTD(Memory Technology Devices)를 위한 파일시스템은 컨트롤러 없이 직접적으로 플래시 메모리에 접근하며, 소프트웨어적인 웨어-레벨링을 수행하기 때문에 파일시스템을 재구성하기 위해서는 스페어 (OOB, Out-Of-Band) 영역이 반드시 필요하다는 점에 유의해야 한다[4][5].

결과적으로 스마트폰 내의 플래시 메모리 분석 시에는 이미지 수집 방법 ((a) 물리적으로 메모리 칩 추출, (b) root 권한 획득 후 파티션 이미징), 파일시스템의 유형, 운영체제의 보안 정책 (암호화) 등에 따라서 다양한 분석 방법을 적용해야 한다. 이를 바탕으로 한 스마트폰 포렌식 수행 절차는 다음 장에서 상세히 설명한다.

III. 스마트폰 포렌식 수행 절차

스마트폰을 분석할 시에는 [그림 1]과 같이 가장 먼저 기기의 상태를 확인해야 한다. 만약 손상된 기기라면, 물리적으로 플래시 메모리 칩을 추출하여 이미징을 시도할 수 있다. 정상 상태의 기기에 대해서는 JTAG 포트의 연결 가능성을 판단한다. JTAG 포트를 연결할 수 있는 경우에는 프로세서(processor)에 연결된 플래시 메모리에 대한 이미징을 수행할 수 있다. 만약 JTAG을 사용할 수 없는 경우에는 운영체제의 루트(root) 권한 획득 가능 여부와 분석의 목적에 따라서 백업(또는 디버깅) 기능을 활용하거나 플래시 메모리에 대한 이미징을 수행한다. 이 때 데이터 수집을 위해서 무료로 공개된 도구나 상용 포렌식 제품(H/W or S/W)을 활용할 수 있다.

백업 또는 디버깅 기능을 이용해서 수집된 데이터는 각 파일 형식 별로 분석할 수 있으며, 만약 플래시 메모리에 대한 이미지를 수집한 경우에는 보다 체계적인 분석 과정을 수행할 수 있다. 한편 eMMC (Embedded MultiMediaCard)와 같이 플래시 메모리와 컨트롤러(FTL(Flash Translation Layer) 기능)가 통합된 블록 장치가 아니기 때문에 메모리 재구성이 필요한 이미지의 경우에는 메타 페이지 또는 스페어 영역을 이용하여 재구성을 시도할 수 있으며, 재구성이 성공한 경우에는 파일시스템의 할당 영역을 분석하여 스마트폰 내의 데이터를 확인한다. 재구성이 실패한 이미지나 파일시스템의 비할당 영역에 존재하는 임의의 메모리 페이지들의 경우에는 본 논문에서 소개하는 '플래시 메모리 페이지 분석'을 수행한다. 추가적으로 소프트웨어 방식의 FTL 위에 파일시스템이 적용된 파티션(MTD+YAFFS 등)에 대해서 nanddump와 같은 도구(플래시 메모리 드라이버에서 제공하는 명령을 사용하여 메모리 페이지와 스페어를 함께 수집)를 사용하지 못하고, dd로 이미징하여 파일시스템을 구성할 수 없는 경우에도 본 논문에서 설명하는 플래시 메모리 페이지 분석 방법을 적용할 수 있다. 이후의 장에서는 단편화된 플래시 메모리 페이지를 분석하는 방법에 대해 상세히 설명한다.



〔그림 1〕 스마트폰 포렌식 분석 과정

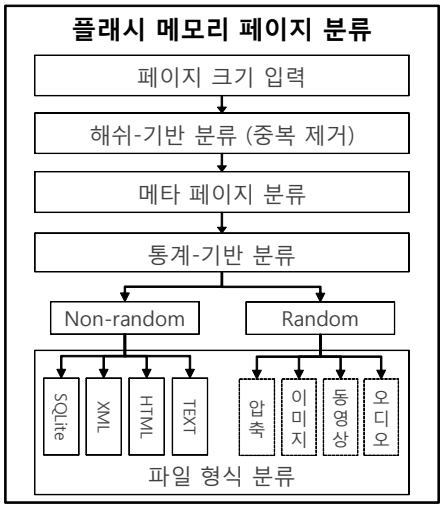
IV. 플래시 메모리 페이지 분석 기법

본 장에서는 플래시 메모리 페이지를 분석하는 기법을 소개한다. 스마트폰에서는 일반적인 PC에 비해서 적은 종류의 파일 형식이 사용되기 때문에 보다 효율적으로 분석을 수행할 수 있다.

4.1 페이지 분류

〔그림 2〕는 단편화된 플래시 메모리 페이지를 분류하는 과정을 나타낸다.

분류에 앞서서 페이지의 크기를 입력해야 한다. 스페어 영역이 함께 추출된 경우에는 보다 수월하게 페이지 크기를 탐지할 수 있을 것이다. 만약, 스페어 영역이 존재하지 않는 경우에는 각 페이지 경계의 변화 정도에 따라서 페이지의 크기를 알아낼 수 있다. 또한 파일시스템이 구성되는 경우에는 쉽게 클러스터 또는 블록의 크기를 확인할 수 있다. 페이지의 크기는 이후의 과정에서 매우 중요한 값이기 때문에 올바르게 입력을 해야 분석의 정확도를 높일 수 있다.



〔그림 2〕 플래시 메모리 페이지 분류 과정

4.1.1 해시 기반 분류

다음으로 해시(Hash) 알고리즘을 이용하여 중복 페이지를 제거하는 작업을 수행한다. 플래시 메모리의 특성상 페이지 내부의 데이터가 수정될 때는 해당 페이지를 직접적으로 수정하지 못하고, 해당 페이지를 삭제하고 다시 쓰거나 새로운 페이지를 할당하여 데이터를 저장한다. 플래시 메모리에서의 삭제(Erase)는 블록(Block) 단위로만 이루어지기 때문에 많은 경우에 이전의 데이터가 남아있게 되어 다수의 중복 페이지가 존재할 가능성이 있다. 따라서 이러한 중복 페이지는 효율적인 분석을 위해서 제거되어야 할 필요가 있다.

한편 동일한 파일이 서로 다른 위치에 정상적으로 존재할 수도 있지만, 중복된 데이터는 효율적인 분석을 방해하기 때문에 이 과정에서 제거될 필요가 있다.

4.1.2 메타 페이지 분류

대표적인 메타 페이지로는 YAFFS의 청크 헤더(Chunk header), RFS와 EXT의 디렉터리 엔트리(Directory entry) 등이 있다. 이 과정을 통해 분류된 메타 페이지로부터 해당 이미지 내에 존재했던 파일 또는 디렉터리의 목록(이름, 시간 정보, 크기 등)을 확인할 수 있다. 경우에 따라서 이미지 내에 메타 페이지가 다 수 존재할 수 있으므로 분석 대상의 크기를 줄이는데 효과적이다.

4.1.3 통계 기반 분류

이번 단계에서는 Random 페이지와 Non-random 페이지를 분류한다. 여기에서 Non-random은 내부 형식이 식별 가능하기 때문에 텍스트, 숫자 등의 값을 쉽게 확인 수 있다. 반면에 Random은 압축되거나 암호화된 경우로 구조를 식별하거나 값을 확인하기 어렵다. 스마트폰에 존재하는 대표적인 Random 데이터로는 이미지 파일(JPEG, PNG, GIF), 멀티미디어 파일(MP3, MP4, AVI), 압축 파일(ZIP, GZIP) 등이 있다. 플래시 메모리 내에서 각 페이지들은 메모리 전역에 흩어져서 존재하므로 Random 데이터를 정확하게 추출하는 것은 매우 어렵다. 따라서 통계 기반으로 Random과 Non-random 페이지를 분류하면 데이터 복구 작업의 효율을 높일 수 있다.

Random 페이지를 분류하기 위해서는 먼저 주요 Random 파일에 대한 시그니처(signature) 검색을 시도해야 한다. 이는 Random 파일의 경우에도 헤더 영역에는 Non-random 데이터가 다수 포함되는 경우가 있기 때문이며, 이 과정을 통해서 주요 Random 파일의 헤더를 분류할 수 있다. 그 후에는 각 페이지에 대해서 포커 테스트(Poker test)와 같은 통계 테스트를 수행하여 Random 페이지를 분류한다〔16〕. 이 때 파일의 마지막 페이지에는 슬랙(Slack) 영역이 존재할 수 있기 때문에 이를 제외한 실제 데이터에 대하여 테스트를 수행해야 한다. 참고로 슬랙 영역을 제외시키기 위해서는 페이지의 마지막 바이트부터 차례로 확인하면서 슬랙 영역의 값(일반적으로 0x00)이 연속되기 시작하는 위치를 계산하여, 해당 위치 이후의 값은 통계 테스트에 포함되지 않도록 한다.

4.1.4 파일 형식 분류

이전 과정을 통해 분류된 Random 페이지(주요 Random 데이터인 압축, 이미지, 동영상, 오디오 등의 파일 형식은 이전 과정에서 분류됨)를 제외한 Non-random 페이지에 대하여 파일 형식 분류를 수행한다. 알려진 파일 시그니처들을 이용하여 분류를 수행하며, 이 과정을 통해 단일 페이지 크기 이내의 파일은 완전하게 복구할 수 있다. 텍스트, XML, HTML, plist 등과 같은 형식의 파일은 그 크기가 작은 경우가 많기 때문에 파일 형식 분류 과정에서 완

〔표 1〕 스마트폰 내의 주요 파일 형식

분류	파일 형식
사용자 데이터 (앱 데이터)	SQLite 데이터베이스, XML, plist (XML or binary)
텍스트	TXT (ASCII,UTF-8,UTF-16 등)
웹 페이지	HTML, JS, SWF 등
압축	ZIP (APK, IPA 포함), GZIP 등
이미지	JPG, PNG, GIF, SVG, BMP 등
동영상	MP4, MOV, AVI, MKV 등
오디오	M4A, 3GA, MP3 등
실행 파일	DEX (Android), Mach-O (iOS) 등
기타	문서 파일 (MS Office, PDF) 등

전한 파일을 획득할 가능성이 있다.

〔표 1〕은 스마트폰 내의 주요 파일 형식을 나타낸다. 스마트폰 운영체제에 따라 주요 데이터를 저장하기 위한 형식이 다를 수 있지만, 대부분의 경우 사용하는 파일 형식이 한정되어 있어서 효율적인 분석이 가능하다.

스마트폰 운영체제에 관계없이 대다수의 응용프로그램에서는 사용자 데이터를 저장하고 관리하기 위해서 SQLite 데이터베이스 파일을 사용하고 있다. SQLite 데이터베이스 파일은 데이터 블록(페이지[1])의 연속으로 구성되며, 블록의 크기는 2^n (n=9~16) 바이트의 범위에서 최초 데이터베이스 생성시에 설정할 수 있다. 한편 SQLite 파일은 헤더(header), 포인터 맵(pointer map), 인덱스 인터널(index internal), 인덱스 리프(index leaf), 테이블 인터널(table internal), 테이블 리프(table leaf), 오버플로우(overflow) 등 다양한 유형의 데이터 블록으로 구성되기 때문에 모든 유형을 고려하여 정확하게 분류될 수 있도록 한다〔6〕.

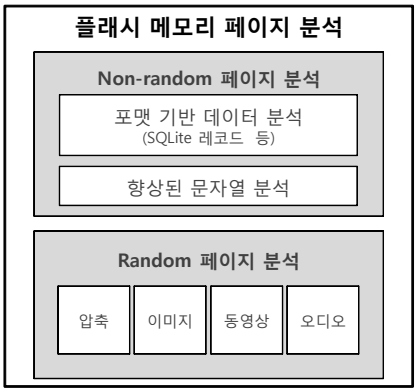
또한 XML(설정 정보), HTML(방문한 사이트) 등은 각 데이터의 시그니처를 활용하여 분류될 수 있도록 한다. 예를 들어 XML이나 HTML은 '〈?xml', '〈!doctype', '〈html' 등의 시그니처를 통해 분류할 수 있다.

한편 〔그림 2〕에서 나타낸 Random 페이지에 대한 파일 형식 분류는 본 논문의 구현 범위에 포함되지 않으며, 향후 연구에서 추가할 계획이다.

1) 플래시 메모리의 페이지와 구분되는 용어로 SQLite 파일 내부적으로 데이터를 저장하기 위해 사용하는 단위(unit)이다.

4.2 페이지 분석

〔그림 3〕은 이전의 과정을 통해 분류된 플래시 메모리 페이지를 분석하는 과정을 나타낸다.



〔그림 3〕 플래시 메모리 페이지 분석 방법

4.2.1 Non-random 페이지 분석

4.2.1.1 포맷 기반 데이터 분석

Non-random 페이지들에 대해서 포맷 기반 데이터 분석을 수행한다. Non-random 페이지에는 일반적으로 가시적인 데이터가 저장되기 때문에 페이지의 조각을 재결합하여 완전한 파일로 만들지 않아도 분석이 가능하다. 이 과정에서는 *bulk_extractor*와 같은 스트림 기반 포렌식 도구가 유용하게 활용될 수 있지만, 현재 지원되는 기능으로는 스마트폰 플래시 메모리 이미지에 적용하기에는 한계점이 있다〔10〕.

특히 대부분의 스마트폰에서는 사용자 데이터를 저장하기 위해서 SQLite를 사용하기 때문에 SQLite 레코드에 대한 분석은 포렌식 관점에서 매우 중요하다. SQLite에는 여러 유형의 데이터 블록이 존재하는데, 그 중에서 테이블 리프(table leaf)에 실제 레코드가 저장된다. 각 레코드를 정확하게 추출하기 위해서는 해당 데이터베이스의 스키마(schema) 정보가 필요하지만, 정상적인 파일이 구성되지 않아 올바른 스키마 정보를 획득할 수 없기 때문에 레코드의 각 컬럼(colume) 정보를 정확하게 확인할 수 없다. 이러한 상황에서 SQLite 레코드를 추출하기 위해 레코드의 저장 구조를 활용할 수 있다. SQLite에서 레코드는 셀(cell) 구조로 저장되는데, 이 정보를 해석하여 각 컬럼의 데이터 형식과 길이 정보를 파악할 수 있다. 이와 같은 방법으로 각 테이블 리프(table

leaf) 데이터 블록으로부터 레코드를 추출할 수 있으며, 추출한 레코드의 컬럼 정보가 동일한 것끼리 분류하여 단편화된 페이지에 흩어져 있는 데이터를 효율적으로 분석할 수 있다[7][8][9]. 한편 각 SQLite 파일에서 사용하는 데이터 저장 단위는 설정에 따라서 최소 512 바이트에서 최대 65,536 바이트의 크기를 가질 수 있는데, 스마트폰 내의 SQLite 파일들이 사용하는 크기가 각기 다를 수 있기 때문에 최소 크기를 기준으로 하여 검색을 수행해야 한다.

위에서 설명한 SQLite 데이터 이외에도 다양한 포맷 기반 데이터가 존재할 수 있으며, 추후의 연구를 통해 분석 기술을 개발할 계획이다.

4.2.1.2 문자열 추출

구조 기반 분석 과정을 수행한 후에 남은 Non-random 페이지에 대해서는 문자열을 추출해서 유의미한 정보를 확인한다. 일반적으로 문자열 추출은 조사 대상 키워드가 존재하지 않을 때 텍스트 기반 데이터에 대한 분석을 위해 실시하는데, 단순한 문자열의 추출은 분석 대상의 양을 증가시킴으로써 비효율적일 수 있다. 따라서 보다 향상된 문자열 추출 알고리즘이 적용될 필요가 있으며, 크게 다음과 같은 4가지 기능이 요구된다.

(1) 다양한 텍스트 인코딩 지원
(ASCII, UTF-8, UTF-16LE, UTF-16BE 등)
(2) 이메일, URL, 전화 번호, IP 주소, 시간 저장 형식 등의 문자 패턴 추출
(3) 문장 단위 추출 (개행 또는 마침표 기준)
(4) 인코딩 (Base64, Base85 등) 된 문자열 자동 디코딩
(5) 중복 제거

*bulk_extractor*는 문자 패턴 추출 기능을 제공하지만[10], 특별한 패턴이 없는 일반 문자열을 추출하는 것도 사용자 데이터 수집 관점에서 의미가 있으므로 현재보다 향상된 문자열 추출 알고리즘을 개발할 필요가 있다. 이와 관련하여서는 본 논문의 범위에 포함되지 않으며, 추후의 연구에서 진행할 계획이다.

4.2.2 Random 페이지 분석

스마트폰에 저장되는 대표적인 Random 데이터로는 압축, 이미지, 동영상, 오디오 등이 있으며, 이러한 Random 데이터가 단편화된 경우에 이를 재구성하거나 일부 조각만을 해석하는 것은 매우 어려운 작업이다. 또한 이미지(또는 비할당 영역) 내의 단편화된 Random 데이터를 분석하기 위해서 전체 데이터 영역을 대상으로 하면, 분석에 소요되는 시간이 오래 걸리면서 정확도가 떨어 질 수 있다. 즉 데이터를 처리하는 범위에 따라서 그 분석의 정확도와 효율성에 큰 영향을 미칠 수 있기 때문에 이전의 과정을 통해서 분류된 Random 페이지만을 대상으로 하면 처리하는 범위를 줄임으로써 효율적으로 작업을 수행할 수 있을 것이다.

현재 단편화된 이미지 데이터를 복구하기 위해서는 *SmartCarving*과 같은 기법을 적용할 수 있다[11][12]. 또한 압축 데이터 분석에는 *ZipRec*과 같은 기법으로 페이지 내에 존재하는 ZIP, PDF 등의 압축 스트림을 복구하거나, deflate 알고리즘으로 압축된 임의의 블록에 대한 복구를 수행해 볼 수 있다. 그러나 현재의 알고리즘은 압축 스트림이 단편화되지 않았다고 가정을 하기 때문에 압축 스트림이 단일 페이지에 저장이 되는 경우에만 복구가 가능하다는 한계점이 있다[13].

Random 페이지에 대한 상세한 분석은 본 논문의 범위에 포함되지 않으며, 추후의 연구를 통해 분석 기술을 개발할 계획이다.

V. 구현 및 실험

5.1 *frag_insight* : 단편화된 데이터 분석 도구

*frag_insight*는 단편화된 페이지(또는 클러스터)에 대한 분석을 위해 개발된 도구이다. [표 2]는 *frag_insight*의 주요 특징을 나타낸다.

[표 2] *frag_insight*의 분석 대상 및 주요 기능

항목	내 용
분석 대상	- 단편화된 플래시 메모리 페이지 - 파일시스템 비할당 영역
주요 기능	[페이지 (또는 클러스터) 분류] - 해시 기반 분류 (중복 제거) - 메타 페이지 분류 (YAFFS, EXT4) - 통계 기반 분류 (압축, 이미지, 동영상, 오디오 헤더 포함) - 파일 포맷 분류 (SQLite, XML, TEXT 등)
	[페이지 (또는 클러스터) 분석] - 구조 기반 데이터 분석 (SQLite 레코드)

*frag_insight*는 단편화된 데이터를 분석하기 위해 개발되었으며, 현재는 스마트폰 플래시 메모리에 대한 분석 기능을 위주로 지원하지만, 향후에는 모든 저장 매체의 비할당(슬랙 포함) 영역을 체계적으로 분석할 수 있도록 기능을 확대해 나갈 계획이다.

*frag_insight*는 현재 개발이 진행 중이며, 오픈 소스 소프트웨어로 'https://github.com/jung-heum/fragmented-data-forensics'에 공개할 예정이다.

5.2 실험 #1 – Motorola Droid 플래시 메모리 이미지 (DFRWS 2011 첼린지 이미지)

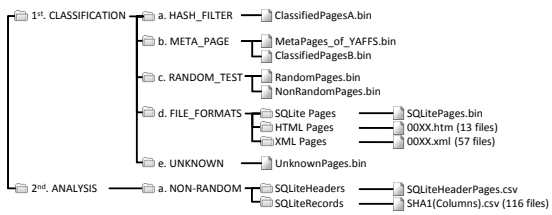
DFRWS 2011 챌린지는 Android 스마트폰의 플래시 메모리 덤프 이미지로부터 사용자의 행위를 파악하는 것이 목적이다[14].

챌린지는 두 개의 시나리오로 구성되며, 시나리오 2의 플래시 메모리 덤프 이미지에는 스페어 영역이 존재하여 YAFFS 내에 존재하는 각 페이지들의 태그(tag) 정보를 확인할 수 있기 때문에 파일 시스템을 재구성(reconstruction) 할 수 있다. 그러나 시나리오 1의 이미지에는 스페어 영역이 존재하지 않기 때문에 파일 시스템을 재구성 할 수 없다. 따라서 시나리오 1을 효율적으로 분석하기 위해서 *frag_insight*를 이용하였다.

실험은 사용자 데이터가 저장되는 /data 파티션의 이미지인 mtdblock6.img(274,464,768 바이트, 약 262MB)를 대상으로 수행하였다. 실험 대상 이미지의 페이지 크기는 2,048 바이트였으며, [그림 4]는 *frag_insight*의 수행 결과를 나타낸다.

[표 3]은 해시-기반 분류를 통해 중복 페이지를 제거한 결과를 보여 준다.

중복 페이지를 제거한 후에는 '분류 결과 A'에 대해서 메타 페이지 분류를 수행한다. 그 결과 [표 4]와 같이 많은 수의 YAFFS의 청크 헤더(chunk header) 페이지가 포함되어 있었으며, 이 과정을 통해 분석 대상의 범위를 줄일 수 있었다.



[그림 4] 실험 #1: *frag_insight* 수행 결과

[표 3] 실험 #1: 해시 기반 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
중복 페이지	36.45MB (38,223,872)	18,664
분류 결과 A	225.30MB (236,240,896)	115,352
전체 페이지 (최초 대상 이미지)	261.75MB (274,464,768)	134,016

[표 4] 실험 #1: 메타 페이지 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
YAFFS 메타 페이지	145.58MB (152,643,584)	74,533
분류 결과 B	79.72MB (83,597,312)	40,819
전체 페이지 (분류 결과 A)	225.30MB (236,240,896)	115,352

메타 페이지 분류 이후에는 '분류 결과 B'에 대해서 통계 기반 분류를 수행한다. 분류 결과 [표 5]와 같이 Random 페이지는 약 2.8MB 였으며, 대다수는 Non-random 페이지였다.

통계 기반 분류 이후에는 'Non-random 페이지'에 대해서 파일 형식 분류를 수행한다. 분류는 파일 시그니처와 내부 형식을 기반으로 수행하며, 분류 결과는 [표 6]과 같다. SQLite 페이지(약 10.2MB)를 비롯하여 다수의 HTML, XML 페이지가 분류되었다.

분류 과정이 완료되면, SQLite 페이지에 대한 상세 분석을 수행한다. [표 7]은 SQLite 페이지에 대한 분석 결과를 나타내며, 116개의 테이블에서

[표 5] 실험 #1: 통계 기반 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
Random 페이지	2.80MB (2,940,928)	1,436
Non-random 페이지	76.92MB (80,656,384)	39,383
전체 페이지 (분류 결과 B)	79.72MB (83,597,312)	40,819

〔표 6〕 실험 #1: 파일 형식 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
SQLite 페이지	10.21MB (10,706,944)	5,228
HTML 페이지	0.03MB (26,624)	13
XML 페이지	0.11MB (116,736)	57
Unknown 페이지	66.57MB (69,806,080)	34085
전체 페이지 (Non-random)	76.92MB (80,656,384)	39,383

16,055개의 레코드가 발견되었다. 각 레코드의 컬럼 정보(INTEGER, TEXT 등)의 조합에 대한 해시 값을 계산하여 각 테이블을 분류하며, CSV 파일(CSV filename = SHA-1(Column1 ‖ Column2 ‖ …. ‖ ColumnN))로 각 테이블을 저장한다. 참고로 동일한 테이블의 레코드라도 NULL 값이 있는 경우에는 서로 다른 테이블로 분류될 수 있으며, 각 테이블에 대한 템플릿을 설정하는 등의 방법으로 이를 해결할 수 있다〔8〕.

〔표 8〕은 추출된 SQLite 레코드의 예를 보여준다.

5.3 실험 #2 - Samsung Galaxy S2 데이터 파티션 (/data) 내의 비할당 영역

두 번째 실험은 삼성 갤럭시 S2 (Android Gingerbread) 내의 데이터를 대상으로 수행하였다. 이 스마트폰의 /data 파티션은 EXT4 파일시스템을 사용하여 데이터를 저장하며, eMMC에 저장되기 때문에 이미지를 획득했을 시에 메모리 페이지의 재구성 과정이 필요하지 않다〔15〕.

〔표 7〕 실험 #1: SQLite 페이지 분석 결과

분류	개수
SQLite 헤더	1,843
SQLite 테이블	116
SQLite 레코드	16,055

〔표 9〕 실험 #2: /data 파티션 내의 파일 정보

조각 개수	설 명	파일 개수
1	설치 데이터 (apk, dex, png, jpg, xml, html, txt 등)	6,323
2	설치 데이터 (SQLite db, apk, dex, png, jpg, gz 등)	265
3-16	사용자 데이터 (SQLite db 등)	74
62	시스템 로그 (/data/log/rtc.log)	1

실험 2에서는 EXT4 파일시스템의 비할당 영역을 대상으로 한다. 실험 대상 스마트폰은 약 8개월 동안 사용한 것으로 /data 파티션은 총 2GB이며, 비할당 영역은 1.36GB이다. EXT4 파일시스템의 할당 영역을 분석한 결과 디렉터리 1,629개와 파일 6,663개가 존재하였다.

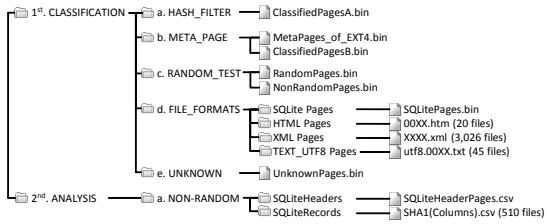
/data 파티션에 저장되는 파일의 저장 상태를 확인한 결과 〔표 9〕와 같이 실제 사용자 데이터가 저장되는 파일들은 여러 조각으로 단편화되어 있었다. 즉 최초 앱(app) 설치 시에 복사되는 파일은 단편화되지 않고 연속적으로 저장되는 경우가 많지만, 사용자 데이터가 저장되는 파일(SQLite 파일 등)은 계속해서 업데이트(수정, 추가, 삭제)되기 때문에 단편화가 많이 발생한다. 따라서 비할당 영역에도 사용자 데이터가 단편화되어 존재할 수 있으며, 단편화된 데이터를 효율적인 분석을 위해서 *frag_insight*를 활용할 수 있다.

실험은 사용자 데이터가 저장되는 /data 파티션의 비할당 영역에 대한 이미지 파일(1,468,866,560바이트, 약1.36GB)을 대상으로 수행하였다. EXT4 파일시스템의 블록 크기는 4,096 바이트였으며, 〔그림 5〕는 *frag_insight*의 수행 결과를 나타낸다.

〔표 10〕은 해시-기반 분류를 통해 중복 페이지를

〔표 8〕 실험 #1: 추출된 SQLite 레코드 예

INT	TEXT	NULL	INTEGER	INT	INT	INT	INT	INT	NULL	TEXT	NULL	INT
1	4124393388	NULL	1304556622728	0	1	255	1	0	NULL	ksmsvzwsms://message/Service Started	NULL	0
1	4124393388	NULL	1304556632613	0	1	255	1	0	NULL	ksmsvzwsms://message/May 4, 2011 8:50:12 PM EDT	NULL	0
1	4124393388	NULL	1304556636370	0	1	255	1	0	NULL	ksmsvzwsms://message/May 4, 2011 8:50:12 PM EDT	NULL	0
1	4124393388	NULL	1304556801665	0	1	255	1	0	NULL	ksmsvzwsms://message/May 4, 2011 8:53:12 PM EDT	NULL	0
1	4124393388	NULL	1304556811634	0	1	255	1	0	NULL	ksmsvzwsms://message/pkg uploaded!	NULL	0



〔그림 5〕 실험 #2: *frag_insight* 수행 결과

제거한 결과를 보여 준다.

〔표 10〕 실험 #2: 해시 기반 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
중복 페이지	1.14GB (1,226,260,480)	299,380
분류 결과 A	0.22GB (242,606,080)	59,230
전체 페이지 (최초 대상 이미지)	1.36GB (11,468,866,560)	358,610

중복 페이지를 제거한 후에는 '분류 결과 A'에 대해서 메타 페이지 분류를 수행한다. 그 결과 〔표 11〕과 같이 31개의 메타 페이지(EXT4의 디렉터리 엔트리)가 포함되어 있었다.

〔표 11〕 실험 #2: 메타 페이지 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
EXT4 메타 페이지	0.12MB (126,976)	31
분류 결과 B	231.25MB (242,479,104)	59,199
전체 페이지 (분류 결과 A)	231.37MB (242,606,080)	59,230

메타 페이지 분류 이후에는 '분류 결과 B'에 대해서 통계 기반 분류를 수행한다. 분류 결과 〔표 12〕와 같이 Random 페이지가 약 50MB 였으며, 그 외에는 Non-random 페이지였다.

통계 기반 분류 이후에는 'Non-random 페이지'에 대해서 파일 형식 분류를 수행한다. 분류는 파일 시그니처와 내부 형식을 기반으로 수행하며, 분류된 결과는 〔표 13〕과 같다. SQLite 페이지(약 3.79 MB)를

〔표 12〕 실험 #2: 통계 기반 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
Random 페이지	50.06MB (52,494,336)	12,816
Non-random 페이지	181.19MB (189,984,768)	46,383
전체 페이지 (분류 결과 B)	231.25MB (242,479,104)	59,199

비롯하여 다수의 HTML, XML, TEXT 페이지가 분류되었다.

〔표 13〕 실험 #2: 파일 형식 분류 결과

분류	전체 사이즈 (bytes)	페이지 개수 (전체 사이즈 / 페이지 크기)
SQLite 페이지	3.79MB (3,977,216)	971
HTML 페이지	0.08MB (81,920)	20
XML 페이지	11.82MB (12,394,496)	3,026
TEXT 페이지	0.18MB (184,320)	45
Unknown 페이지	165.32MB (173,346,816)	42,321
전체 페이지 (Non-random)	181.19MB (189,984,768)	46,383

분류 과정이 완료되면, SQLite 페이지에 대한 상세 분석을 수행한다. 〔표 14〕는 SQLite 분석 결과를 나타내며, 510개의 테이블에서 17,233개의 레코드가 발견되었다.

〔표 14〕 실험 #2: SQLite 페이지 분석 결과

분류	개수
SQLite 헤더	32
SQLite 테이블	510
SQLite 레코드	17,233

〔표 15〕는 추출된 SQLite 레코드의 예를 보여주고, 〔표 16〕은 정상 영역에 존재하는 webview-Cache.db 파일의 cache 테이블을 SQLite 뷰어로 확인한 내용의 일부를 나타낸다. 이를 통해 〔표 15〕에 나타낸 레코드가 정상적으로 복구된 것임을 확인할 수 있다.

VI. 향후 연구

향후에는 데이터의 단편화에 대응하기 위해서 플래시 메모리 페이지(파일시스템의 클러스터 또는 블록)를 정밀하게 분류하는 연구를 수행할 것이다. 특히 데이터의 유형을 판별하기 위한 시그니처가 없거나 둘 이상의 서로 다른 데이터가 하나의 페이지에 존재하는 등의 다양한 경우에 적용이 가능한 분류 기법을 개발할 것이다.

나아가서 Random 페이지에 대한 분류를 암호화, 데이터 압축, 이미지, 동영상, 오디오 등으로 세분화할 수 있다면 효율적인 분석에 도움이 될 것이다. 또한 단편화된 Random 페이지들로부터 유의미한 데이터를 획득하기 위한 연구를 수행할 것이다. 최근까지는 단편화된 이미지(그래픽) 파일에 대해서 연구가 주로 이루어져 왔지만, 데이터 압축이나 동영상 등으로 연구를 확장시킬 필요가 있다.

현재 버전의 *frag_insight*는 스마트폰의 플래시 메모리를 주요 대상으로 하여 개발 중이지만, 추후에는 이를 확장하여 다양한 저장 매체에서 범용적으로 활용이 가능하도록 할 계획이다.

VII. 결 론

디지털 포렌식 분야에서 저장 매체 내의 데이터를 분석하기 위한 연구는 매우 활발하게 진행되어 왔다. 파일시스템 분석, 비할당 영역의 삭제된 데이터 복구 등 많은 기법이 연구 및 개발 되었으며, 실제 디지털 포렌식 조사 과정에서 널리 활용되고 있다.

최근 스마트폰의 보급과 함께 다양한 앱(app)이 사용되면서 민감한 개인 정보들이 기기 내의 플래시 메모리에 저장되고 있다. 현재 스마트폰 기기별로 데이터를 수집하고 분석하는 방안이 연구되어 조사를 수행하고 있지만, 데이터 수집 방법에 따라서는 파일시스템을 제대로 구성할 수 없어서 분석이 어려운 경우가 존재할 수 있다. 또한 파일시스템 분석이 가능한 경우에도 사용자가 특정 데이터 또는 앱을 삭제한 경우에 비할당 영역으로부터 해당 데이터를 추출 및 분석하는 것에 많은 어려움이 있는 것이 사실이다.

헤더-푸터, 파일 크기, 파일 형식 등에 기반한 복구(카빙) 기법은 데이터가 연속적으로 저장되는 경우를 가정하기 때문에 단편화가 많이 발생하는 환경에서는 적용하는데 한계가 있다. 특히 최근에 널리 사용되는 스마트폰에서는 사용자 데이터 저장을 위해서 SQLite와 같은 데이터베이스를 이용하는데, 새로운 데이터가 계속해서 추가되면서 단편화가 많이 일어나게 된다.

본 논문에서는 단편화된 페이지(파일시스템의 클러스터 또는 블록)를 분류하고 분석하는 기법을 소개하였다. 소개한 기법은 파일시스템 구성이 어려운 플래시 메모리 덤프 이미지를 분석하는데 활용될 수 있다. 또한 파일시스템 분석이 가능한 경우에는 비할당 영역에 대해서 연속적인 데이터 복원(카빙)을 수행한 후에

〔표 15〕 실험 #2: 추출된 SQLite 레코드 예

TEXT	TEXT	TEXT	TEXT	INT	TEXT	TEXT	TEXT	INT	NULL	INT	NULL	NULL
http://search.4shared.com/css/ajax-suggestions.css	7d73f99a	Tue, 03 Jan 2012 04:55:52 GMT	W/"877-1325566552000"	1326161262457	Tue, 10 Jan 2012 02:07:29 GMT	text /css		200	NULL	877	NULL	NULL
http://blogimgs.naver.net/mobile/u_cm2.png	4b116206	Fri, 09 Dec 2011 08:38:30 GMT	"d3e-b59fcd80"	1357277950563	Fri, 04 Jan 2013 05:38:58 GMT	image /png		200	NULL	3390	NULL	NULL
http://mail.korea.ac.kr/t_default/ko/images/leftmenu/bgMenu_topTop_new.gif	cd4ae7d3	Fri, 26 Feb 2010 13:28:32 GMT	"5376d-1427-48080e02c2000"	1319352063745	Sun, 23 Oct 2011 06:40:46 GMT	image /gif		200	NULL	5159	NULL	NULL

〔표 16〕 실험 #2: webviewCache.db 파일의 cache 테이블 내용 일부

url	filepath	lastmodify	etag	expires	expiresstring	mime-type	en-cod-ing	httpst-atus	locat-ion	conten-tlength	con-tentdis-position	crossdo-main
http://www.4shared.com/css/ajax-suggestions.css	e44a5c6f	Tue, 03 Jan 2012 04:55:52 GMT	W/"877-1325566552000"	1326161248768	Tue, 10 Jan 2012 02:07:15 GMT	text /css		200	NULL	877	NULL	NULL
http://static.4shared.com/images/logo.png	880d3ecc	Mon, 17 Jan 2011 04:29:04 GMT	W/"1850-1295238544000"	1314352448125	Fri, 26 Aug 2011 09:54:07 GMT	image /png		200	NULL	1850	NULL	NULL
http://static.4shared.com/images/4slogo_150x40.gif	7343258d	Mon, 17 Jan 2011 04:29:02 GMT	W/"2311-1295238542000"	1314352448116	Fri, 26 Aug 2011 09:54:07 GMT	image /gif		200	NULL	2311	NULL	NULL

남겨진 '순수 비할당 영역'을 체계적으로 분석하기 위해서 본 기법을 활용할 수 있다.

데이터의 단편화는 디지털 포렌식 분야에서 큰 장애물 중의 하나이며, 반드시 해결해야 할 과제이다. 이를 해결하기 위해 본 논문에서는 단편화된 데이터를 체계적으로 분석하기 위한 의미 있는 연구를 진행하였다. 이러한 연구 결과는 향후 디지털 포렌식의 연구 분야를 넓힘과 동시에 실제 조사 과정에서 유용하게 활용될 수 있을 것이다.

참고문헌

〔1〕 YAFFS(Yet Another Flash File System), http://www.yaffs.net.

〔2〕 D. Woodhouse, S. Hill, nanddump: dumps the contents of raw NAND chips or NAND chips, 2000. http:// git.infradead.org/mtd-utils.git/blob_plain/HEAD:/nanddump.c.

〔3〕 J. Bédrune, J. Sigwald, iphone-dataprotection, http://code.google.com/p/iphone-dataprotection/.

〔4〕 Memory technology device, http://www.linux-mtd.infradead.org.

〔5〕 Samsung Electronics, LinuStoreii Samsung Onenand Flash Linux Device Driver - GPL Compliance, July. 2008.

〔6〕 SQLite Database File Format, http://www.sqlite.org/fileformat2.html.

〔7〕 Sausage Factory, An analysis of the record structure within SQLite databases, http://forensicsfromthesausa-gefactory.blogspot.com/2011/05/analysis-of-record-structure-within.html, May. 2011.

〔8〕 I. Pooters, P. Arends, S. Moorrees (Fox-IT), "Extracting SQLite records - Carving, parsing and matching," July. 2011. http://www.dfrws.org/2011/challenge/results.shtml.

〔9〕 S. Jeon, J. Bang, K. Byun, S. Lee, "A recovery method of deleted record for SQLite database," Personal and Ubiquitous Computing, vol. 16, no. 6, pp. 707-715, 2012.

〔10〕 S. Garfinkel, "Digital media triage with stream-based forensics and bulk extractor," http://afflib.org/software/bulk_extractor, June 2011.

〔11〕 H. Sencar, N. Memon, "Identification and recovery of JPEG files with missing fragments," Digital Investigation, vol. 6, supplement, pp. S88-S98. 2009.

〔12〕 Adroit Photo Forensics 2011, http://digital-assembly.com/products/adroit-photo-forensics/.

〔13〕 R. Brown, "Reconstructing corrupt DEFLATEd files," Digital Investigation, vol. 8, supplement, pp. S125-S131. 2011.

〔14〕 DFRWS 2011 Forensics Challenge, http://www.dfrws.org/2011/challenge/.

〔15〕 Samsung Electronics, eMMC, http://www.samsung.com/global/business/semicond-uctor/product/flash-emmc/overview.

〔16〕 D. E. Knuth, The Art of Computer Programming, vol 2, Seminumerical Algorithms, Addison-Wesley, 3rd edition, 1997.

〈著者紹介〉

박 정 흠 (Jungheum Park) 학생회원
2007년 2월: 한양대학교 정보통신대학 컴퓨터전공 공학사
2007년 3월~2009년 2월: 고려대학교 정보경영공학전문대학원 공학석사
2009년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 디지털 포렌식, 안티-안티 포렌식

정 현 지 (Hyunji Chung) 학생회원
2010년 2월: 고려대학교 컴퓨터정보학, 산업시스템공학 공학사
2010년 3월~2012년 2월: 고려대학교 정보보호대학원 공학석사
2012년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 디지털 포렌식

이 상 진 (Sangjin Lee) 종신회원
1987년 2월: 고려대학교 수학과 학사
1989년 2월: 고려대학교 수학과 석사
1994년 8월: 고려대학교 수학과 박사
1989년 10월~1999년 2월: ETRI 선임 연구원
1999년 3월~2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월~현재: 고려대학교 정보보호대학원 교수
2008년 3월~현재: 고려대학교 디지털포렌식연구센터 센터장
〈관심분야〉 디지털 포렌식, 심층 암호, 해쉬 함수

손 영 동 (Youngdong Son) 종신회원
1986년: 한국경제신문 정보통신전문기자
2003년: KTH 상무이사/상임감사
2008년: 국가보안기술연구소 소장
2011년: 숭실대학교 IT정책경영학(박사)
2011년~현재: 고려대학교 정보보호대학원 초빙교수
〈관심분야〉 사이버 테러, 사이버전, 사이버 심리전