공학석사 학위논문

디지털 증거 획득과 교환을 위한 포렌식 파일 포맷 설계 및 구현

2009년 2월

부 경 대 학 교 대 학 원

정보보호 협동과정

박 태 식

공학석사 학위논문

디지털 증거 획득과 교환을 위한 포렌식 파일 포맷 설계 및 구현

지도교수 신상욱

이 논문을 공학석사 학위논문으로 제출함.

2009년 2월

부 경 대 학 교 대 학 원

정보보호 협동과정

박 태 식

박태식의 공학석사 학위논문을 인준함.

2009년 2월 25일

주심 이학박사 이 경 현 (인)

위원 공학박사 송 하 주 (인)

위원 이학박사 신 상 욱 (인)

차 례

그림 차례

표 차례

Design and Implementation of Forensic File Format for
Acquisition and Exchange of Digital Evidence

Tae Sick Park

Interdisciplinary Program of Information Security, The Graduate School,
Pukyong National University

Abstract

Computer forensics typically deals with large amount of data. Modern commodity hard disks approach the Terabyte domain, leading to many problems in managing and archiving bit for bit images.

However, most of image file formats can handle only a few giga bytes of disk storages. Also they don't consider protecting the confidentiality of imaged data. In modern computing environment, it is usual that even a personal user uses a storage disk with the capacity of more than a few hundreds of giga bytes. And also, a law court tends to protect the privacy of the suspected until a juridical sentence is done.

Therefore this thesis propose a new forensic image format which can handle high capacity disk storages. The proposed forensic file format supports metadata that can be defined and extended by users. Especially, it has flag field in which users set the encryption and digital signature algorithms.

And for the purpose of promotion of international assistance for digital evidence investigation and transportation of digital data including evidence acquired remote place, exchange format of disk imaging file is proposed. The proposed disk image exchange format can increase digital forensic tool's interoperability.

I. 서 론

1. 연구배경

디지털 포렌식(Digital Forensics)은[1] 컴퓨터, 인터넷 관련 범죄의 폭발적인 증가와 이를 이용하는 수법이 갈수록 다양해져 감에 따라 이러한 범죄유형에 대응하기위해 1980년대 중반부터 디지털 증거 보존, 신원확인, 문서를 다루는 것에서 시작하여 현재까지 지속전인 연구와 발전을 하고 있다. 현재는 법집행기관과 군사기관에 정보수집과 수사의 주요 기술로 인정받고 있으며 또한 핸드폰, PDA, 컴퓨터, 메모리 등의 디지털 매체를 이용한 일반적인 범죄에서도 중요 증거들을 찾아내고 조사하는데 유용하게 사용되고 있으며 첨단기술 유출방지, 지적재산권보호 등 일반기업의 적극적인 활용에 의해 국가와 기업의 재산과 권익을 보호하는데 크게 기여하고 있다.

디지털 포렌식은 간단히 수사준비, 증거획득, 보관이송, 검증 분석, 조사보고 등으로 나눌 수 있으며 각 단계에 대한 연구개발을 바탕으로 지속적인 발전을 하고 있지만 여전히 디지털 매체를 이용한 범죄 수사에 과학적이고 체계적인 개선이 요구되고 있다. 더욱이 상용 도구(ex. EnCase, AFF, FTK)와 오픈 소스 도구 등[1][3][8][9] 다양한 도구들이 디지털 포렌식을 지원하지만 대표적인 도구를 제외하고는 범죄수사에서 법적으로 인정을 받지 못하고 있는 실정이다.

법적으로 인정받기위한 디지털 포렌식 단계 중 가장 초기단계는 증거 획득 단계라 할 수 있다. 따라서 본 논문에서는 증거 획득 단계

에 디지털 증거를 좀 더 편리하고 효율적이며 증거력을 가지는 디지털 증거획득에 대한 방법을 제시한다.

또한 현재 상용 도구와 오픈소스 도구 등 모든 포렌식 수집도구들은 각각의 획득 포맷을 가지고 있음으로 인해 특정 분석 도구에 의존적으로 수사가 제한된다. 하지만 사이버 범죄의 특성상 원격지에서 수집한 데이터에 대해 증거 분석을 실행해야 하는 경우가 발생할 수 있으며, 분석된 증거의 신뢰성을 위해 수집된 데이터를 이용하여 다양한 분석이 요구되어질 수 있으므로 이러한 증거 교환 또는 전송의 경우 증거의 증거력이 회손 되지 않고 효율적이고 안전하게 교환할 수 있는 방안을 제시한다.

2. 연구 내용 및 구성

본 논문은 디지털 포렌식의 수사준비, 증거획득, 보관이송, 검증분석, 조사보고의 단계 중 증거획득과 보관이송 단계에서 증거력을 가지는 디지털 증거획득과 증거 교환을 할 수 있는 프로토타입(Prototype)설계하고 구현한다.

쉽게 접할 수 있는 포렌식 오픈 소스 툴을 본 논문에서 제안하는 증거획득 포맷과 증거 교환 포맷에 맞게 개선하여 포렌식의 증거획득 단계와 보관이송 단계에서 활용 가능한 툴을 개발한다.

본 논문의 구성은 다음과 같다. 1장인 서론에 이어 2장에서는 기존의 상용도구와 오픈소스 도구의 파일 포맷을 분석하고, 3장에서는 오늘날 대용량 매체가 일반적인 것과 앞선 2장에서 분석한 포맷들의

효율성과 장점을 가춘, 본 논문에서 제안하는 대용량 저장 매체 포렌식 획득 툴 파일포맷에 대하여 다룬다. 4장에서는 본 논문에서 제안한 이미징 포맷이외에 상용 툴, 오픈 소스 툴등으로 획득된 증거 이미지의 증거력을 잃지 않고 효율적이고 안전하게 이미지의 교환 또는 전송이 이루어질 수 있는 디스크 이미지 교환 포맷에 대해 다룬다. 5장에서는 3장, 4장에서 제안하고 정의한 포맷들을 가지는 프로토타입의 구성과 구현 그리고 개발된 툴 테스트에 관한 내용을 다루고 마지막 6장에서 결론을 맺는다.

II. 관련 연구

증거로 사용되는 디지털 데이터는 대개 전문적이고 공개되지 않은 포맷으로 저장된다. 이들 포맷들은 보통 증거에 관한 메타 데이터를 포함한다. 공개되지 않은 포맷은 데이터에 사용될 수 있는 툴과 분석 기술을 제한한다.

이장에서 기존 데이터 획득 툴들의 이미지 파일 포맷을 분석하고, 대용량 저장장치의 데이터 획득과 관리가 가능한 이미지 파일 포맷을 다음 장에 제안한다.

1. 기존 데이터 획득 포렌식 툴 파일 포맷 분석

가. 포맷개요

여기서는 각 포맷의 간략한 개요를 제공한다. 먼저 AFF(Advanced Forensic Format)[3]은 Simson Garfinkel and Base Technology에 의해 제안되었고, 3가지 변형, AFF, AFD, AFM이 있다. AFF는 하나의 파일에 모든 데이터와 메타 데이터를 저장하고, AFD는 여러 개의 작은 파일에 데이터와 메타 데이터를 저장한다. AFM은 raw 포맷으로 데이터를 저장하고 개별적인 파일로 메타 데이터를 저장한다.

DEB(Digital Evidence Bag)을 사용하는 두 가지 포맷이 있다. 첫 번째 포맷은 Philip Turner와 Qinetiq에 의해 제안되었으며, 포맷은 공개되었다. 이 포맷은 증거와 관련 메타 데이터를 저장하기 위해

여러 개의 파일들을 사용한다. 메타 데이터는 ASCII 파일로 저장되며, 이 포맷을 위한 툴은 공개적으로 배포되지 않았다. 두 번째 DEB 포맷은 Webston Technology에 의해 제안되었다. 이 포맷은 증거와 메타 데이터를 저장하기 위해 XML을 사용한다. 이 포맷은 Air Force Research Labs에서 연구용으로 개발되었다.

Encase 포맷은[8][9] Guidance Software에서 자신들의 Encase 툴에서 사용하기 위해 정의된 공개되지 않은 포맷이다. 이 포맷은 하드 드라이브 이미지와 개별 파일들을 저장하기 위해 사용된다. 이 포맷의 전임 포맷은 Expert Witness 포맷이고, Encase 포맷은 Expert Witness 포맷에 새로운 메타 데이터를 추가했다.

Gfzip(Generic Forensic ZIP) 포맷은 Rob J Meijer에 의해 제안되었고 AFF와 유사한 데이터 구조를 가지며 공개된 포맷이다. Gfzip 파일은 메타 데이터가 증거 데이터 이후에 저장되도록 raw 포맷과 호환 가능하다. 또한 redundant 데이터 블록들이 저장되지 않는 'packed' 모드를 제공한다.

iXImage 포맷은 iLook 툴에 의해 사용되며 U.S. Internal Revenue Service(IRS)에 의해 개발되었고 법 집행 기관과 정부에서만 사용된다.

ProDiscover 포맷은 하드 드라이브 이미지를 저장하기 위해 Technology Pathways에 의해 정의되었다. 이 포맷은 공개되었고 공개된 문서가 있다.

Raw 포맷은 저장될 필요가 있는 정확한 데이터를 포함한 단순한 파일이고, 하드 디스크 섹터, 파일, 네트워크 패킷 등의 임의의 데이터 유형을 포함할 수 있다. Raw 파일은 임의의 툴에 의해 쉽게 생성되고 읽을 수 있지만, 메타 데이터를 저장하지 않고 압축되지 않는다.

ASR Data에 의해 정의된 두 가지 SMART 포맷이 있다. 디폴트 포맷은 메타 데이터를 분리된 개별 텍스트 파일(그 내용은 쉽게 조사될 수 있지만 정확한 layout은 공개되지 않았다)에 저장한다. 두 번째 포맷은 SMART Expert Witness Compression 포맷으로 불리고, Expert Witness 포맷에 기반 하여 정의된다.

(1) 기존 데이터 획득 포렌식 툴 파일 포맷

(가) dd 이미지

- raw 미디어 데이터의 단순한 복사본이다.
- 이들 이미지들은 좀더 다루기 쉽도록 하기 위해 좀더 작은 chunk로 분할된다. 그리고 결과적인 이미지들은 FAT 또는 DVD/CDROM과 같은 제한된 파일시스템과 매체에 일치될 수 있다.
- 장점
  - dd 포맷은 매우 간단하고 이미지를 다루기 위해 간단한 툴을 사용할 수 있다. 즉 재분할하기 위해 표준적인 *split* 명령을 사용할 수 있다.
  - 필요한 만큼 이미지를 분할하고 재분할하는 것이 가능하다. 예를 들면 큰 이미지를 획득한 후 DVD로 재분할하는 것이 필요할 때 중요하다.
  - pipe로 write 될 수 있다. streamable 출력은 이미지가 획득될 때 dd 이미지를 netcat 또는 ssh로 piping 하는 것이 쉽게 수행될 수 있는 간단한 전송을 허용한다.
- 단점
  - dd 이미지는 압축이 적용되지 않기 때문에 매우 커질 수 있다. dd

이미지가 Zip 또는 gzip과 같은 일반적인 툴을 사용하여 압축된다면, 그것들에 직접적으로 동작하는 것이 불가능하다. 이것은 주로 전통적인 스트림 압축기가 파일에서 임의의 위치를 찾는 것이 불가능하다는 사실 때문이다. 큰 크기 때문에 FAT과 같은 작은 파일 시스템에 적응시키는 것이 불가능하다.

- 파일 이름 외의 메타 데이터가 없다. 파일 이름이 손실되면 파일에 어디에 속하는지 아는 것이 불가능하다. 메타 데이터에 기록되는 MD5/SHA1 해쉬가 없다.
- 파일에 관한 checksum이 없다. dd 이미지는 손상에 대해 낮은 강건성(robustness)을 가진다. 보통 두 가지 형태의 손상이 발생할 수 있다.
  ① 세그먼트 손실 : 분할된 집합에서 dd 이미지들 중의 하나가 손실될 때 발생 가능(즉 긁힌 CD/DVD).
  ② overwrite 되어진 데이터 : 몇 가지 파일이 overwrite 되어져 파일에서 “hole”을 생성한다.
- 위의 1번의 경우, 손실된 세그먼트가 얼마만큼의 크기를 사용했는지를 안다면 ‘0’으로 채워진 새로운 세그먼트가 만들어져 이미지에 삽입될 수 있다.
- 손상 유형 2번의 경우는 파일에서 작은 블록들에 관한 checksum을 가지지 않기 때문에 복구될 수 없다. 전체 이미지의 해쉬를 계산하여전체 이미지가 손상되었는지를 결정할 수 있지만 정확하게 어느 블록이 손상되었는지를 알지 못한다.

(나) EWCF(Expert Compression Format) spec

– 압축된 데이터를 저장하기 위한 산업 표준 포맷

– EWCF는 전체 데이터 스트림을 압축 해제 하지 않고 압축되지 않은 데이터에서 임의의 오프셋(offset)을 사용자가 액세스하는 것을 허용한다.

– 하나 이상의 세그먼트 파일들에 하나의 이미지를 저장 가능

– 각 파일은 표준 13-바이트 헤더, 일련의 섹션(section)들로 구성

– 섹션은 연속적으로 정렬된다.

– 세션은 두 개의 파일에 걸쳐 저 장될 수 없다.

· chunk : 32k 데이터 런(64개의 표준 섹터들)

· 모든 오프셋은 세그먼트파일의 시작에 상대적이다.

① 파일헤더

– 각 파일은 13-바이트 헤더로 시작

[표1] 8 바이트의 서명 부분(Signature part)

Bytes	3	1	1	1	1	1
Data	"EVF"	0x09	0x0d	0x0a	0xff	0x00

[표2] 5 바이트의 필드 부분(Fields part)

Bytes	1	2	2
Data	0x01	1 또는 그 이상	0x0000
Meaning		세그먼트 번호	

② 섹션

– 각 섹션은 동일한 표준 데이터로 시작하고, 다음의 레이아웃을 가진다.

[표 3] 섹션 레이아웃

Offset	Bytes	Data	Meaning
0 (0x0)	16	"volume", "header" 등	섹션 유형 스트링

16 (0x10)	8		현재 파일에서 다음 섹션까지의 64-비트 오프셋
24 (0x18)	8		섹션의 64-비트 바이트 크기
32 (0x20)	40	0x00...	패딩
72 (0x48)	4		모든 이전 섹션 데이터의 CRC

③ 섹션 유형

- EWCF는 다음의 섹션 유형들을 사용한다.

  . header, volume, table, next, done

- 이들 섹션 유형들 중의 몇몇은 위의 표준 섹션 구조 직후에 시작하는 유일한 데이터를 가진다.

- 'header' 섹션

[표 4] header 섹션

Offset	Bytes	Data	Meaning
76 (0x4c)	섹션의 끝까지	zlib 압축된 데이터	Comments 구조

. Comments 구조는 다음의 tab, newline, delimited 포맷의 텍스트 스트링이다(각 셀에서 데이터는 tab 문자에 의해 분리되고 각 행은 newline 문자에 의해 구분된다).

. 첫 번째 3개의 라인은 표준이고 변경되지 말아야 한다.

. 세 번째 라인에서 문자들은 4번째 라인에서 필드들의 내용에 대한 암시(reminder)로 작용한다.

[표 5] Comments 구조

1								
main								
c	n	a	e	t	m	u	p	r
Case Number	Evidence Number	Unique Description	Examiner Name	Notes	Acquired Date	System Date	pwhash	char

. Case Number, Evidence Number, Unique Description, Examiner Name, Notes는 자유 형태 (tab EH는 newline 문자들을 포함하지 말아야 한다)이다.

. Acquired Date, System Data 는 “2002 3 4 10 19 59” 의 형태이다(March 4, 2002 10:19:59).

. pwhash는 문자 ‘0’ 이어야 한다.

. char는 ‘b’, ‘f’, ‘n’ 3개의 문자들 중의 하나이어야 한다. 이것은 각각 “best”, “fastest”, “no compression” 을 나타낸다. EWCF는 “f” 를 사용한다.

. header 섹션은 첫 번째 세그먼트 파일에서만 나타나야 한다.

– ‘volume’ 섹션

[표 6] volume 섹션

Offset	Bytes	Data	Meaning
76 (0x4c)	4	1	Reserved
80 (0x50)	4		Chunk Count
84 (0x54)	4	64	Sectors per Chunk
88 (0x58)	4	512	Bytes per Sector
92 (0x5c)	4		Sector Count
96 (0x60)	20	0x00...	Reserved
116(0x74)	45	0x00...	Padding
161(0xa1)	5		Reserved (Signature)

166(0xa6)	4		오프셋 76에서 시작하여 모든 이전 'volume' 데이터들의 CRC

. volume 섹션은 첫 번째 세그먼트 파일에서만 나타나야 한다.

- 'table' 섹션

[표 7] table 섹션

Offset	Bytes	Data	Meaning
76 (0x4c)	4	1	Chunk Count (for this table)
80 (0x50)	16	0x00...	Padding
96 (0x60)	4		오프셋 76에서 시작하여 모든 이전 'table' 데이터들의 CRC
100 (0x64)	필요한 만큼		오프셋 배열(offset array)
오프셋 배열의 끝으로부터	섹션의 끝까지		zlib 압축된 데이터 Chunks

. 오프셋 배열은 일련의 연속적인 4-바이트 unsigned integer 값들이다. 각 항목은 압축된 Chunk'의 시작에서의 오프셋이다.

. 각 값들의 최상위 비트는 1로 셋팅 되어야 한다. Chunk 당 하나의 항목이 있어야 한다.

. 각 table 섹션은 16375 항목들을 유지할 수 있다. 더 많은 항목들이 필요하다면, 파일 당 복수개의 table 섹션을 생성해야 한다.

- 'next'와 'done' 섹션

. 각 파일은 'next' 또는 'done' 섹션으로 끝난다.

. 각 파일이 EWC 이미지에서 마지막 세그먼트라면, 섹션은 'done' 이름을 가질 것이다.

. 그렇지 않다면 섹션은 또 다른 세그먼트 파일이 읽어져야 한다는 것을 지시하기 위해 'next' 이름을 가질 것이다.

. 파일에서 마지막 섹션이기 때문에 이들 섹션 유형에 대한 "next section" 포인터는 섹션 자체의 시작을 가리킨다.

. 이들 섹션 유형들은 유일한 데이터를 가지지 않는다.

④ CRC 알고리즘

- 다음의 함수는 EWCF에서 사용되는 4-바이트 CRC 값을 반환한다.
- 새로운 CRC를 시작하기 위해 함수를 호출하는 경우, 세 번째 인자(prevkey)는 '1' 이어야 한다.

```
uint Expert_Witness_Compression_crc(void *buffer, int buffersize, uint prevkey)
    {       unsigned char *cbuffer = (unsigned char *)buffer;
            unsigned int b = prevkey & 0xffff;
            unsigned int d = (prevkey >> 16) & 0xffff;

            for(int i=0; i < buffersize; i++)
            {       b += cbuffer[i];
                    d += b;
                    if ( i != 0 && ( (i % 0x15b0 == 0) || (i == buffersize - 1) ) )
                    {       b = b % 0xfff1;
                            d = d % 0xfff1;
                    }
            }
            return (d << 16) | b;
    }
```

⑤ 장단점 분석

– 장점

. 이미지를 작은 파일들로 분할할 수 있다.

. 메타 데이터가 파일과 함께 저장된다. 그러므로 세그먼트 순서를 맞추기 위해 파일 이름을 필요로 하지 않는다.

. 압축을 지원한다. 압축은 이미지 내에서 빠른 검색을 위해 블록 기반(보통 32kb)으로 수행된다. 대부분의 패키지들은 이미지 파일에서 직접 동작한다.

. 각 압축된 블록에 checksum을 적용하여 손상에 대해 개별 블록을 검사하는 것이 가능하다. 이것은 이론적으로 몇몇 경우에 손상에도 불구하고 파일에 동작을 계속하는 것을 가능하게 한다. Encase 자체는 이것을 지원하지 않는다. 다른 포렌식 패키지들은 파일 포맷의 강건한 처리를 지원하고 몇몇 손상으로부터 복구할 수 있다.

– 단점

. 이미지를 다루기 위해 간단하고 표준적인 툴을 사용할 수 없다. 즉 이미지를 더 분할하기 위해 *split* 명령을 사용할 수 없다.

. 전체 이미지를 압축해제하여 재 압축하지 않는다면, 한번 획득되어지면 이미지를 더 작은 파일들로 재분할하는 것이 불가능하다.

. streamable 출력에 write 할 수 없다. EWF는 출력 스트림을 찾기 위해 writer를 요구한다. 이것은 netcat 또는 ssh에 직접 출력을 pipe 하는 것이 불가능하다.

. 하나의 이미지를 2GB 이상의 크기로 생성하는 것이 불가능하다. 이 제약 사항은 파일 내에서 32비트 오프셋의 사용으로부터

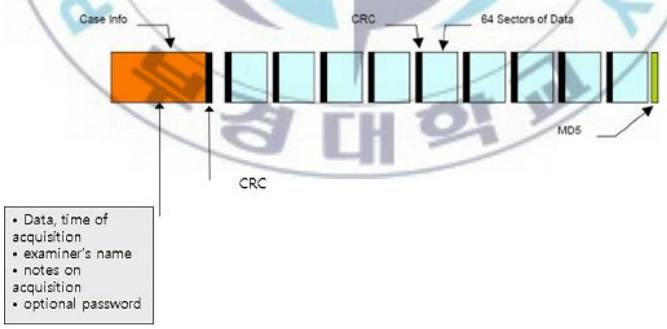
기인한다.

. 오류로부터 복구하기가 어렵다. 압축된 블록 데이터가 overwrite 되어지는 손상 2의 경우로부터 복구하는 것이 가능할 수도 있지만, 인덱스가 overwrite 되어지면 훨씬 더 많은 데이터가 손실된다. 예를 들면 1000 바이트의 overwrite는 섹터 영역에서 하나의 손실 블록을 야기하지만 table 섹션을 overwrite 한다면 250*32kb = 8Mb가 손실된다.

. 거의 독점적인 포맷. 파일에 관한 많은 것들이 공개적으로 알려지지 않는다.

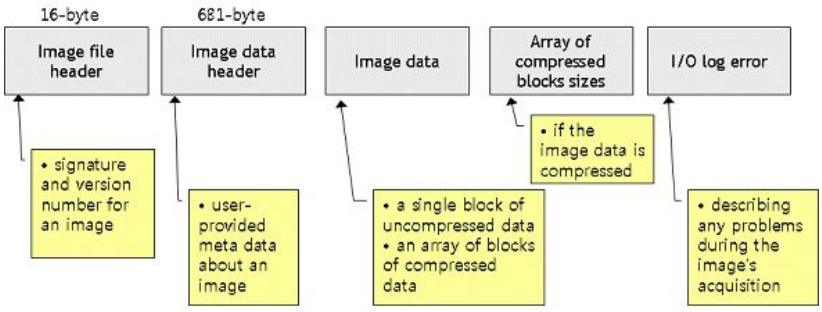
(다) Encase와 ProDiscover 포맷

① Encase 포맷



[그림 1] Encase 포맷

② ProDiscover 포맷



[그림 2] ProDiscover 포맷

(라) AFF(Advanced Forensic Format)

– 목적

. 압축을 하거나 또는 하지 안르고 디스크 이미지를 저장 가능

. 임의의 크기 디스크 이미지를 저장가능

. 디스크 이미지 내에 또는 개별적으로 분리된 메타 데이터를 저장 가능

. 사용자 정의 name/value 쌍의 임의의 메타 데이터 사용가능

. 확장 가능성

. 단순한 설계

. 다중 플렛폼, 오픈 소스 구현

. 지적 재산권 제한으로부터 자유로움

. 내부의 self-consistency 검사 제공. 이미지의 일부분이 손상되거나 손실되더라도 이미지의 다른 부분이 복구될 수 있다.

. 전통적인 해쉬 함수와 X.509 인증서에 기반한 진보된 전자 서명으로 증거파일의 확실성 인증을 제공

- AFF 계층

. AFF는 두 개의 계층으로 분할

- disk-representation 계층 : 디스크 이미지와 이에 관련된 메타 데이터를 저장하기 위해 사용되는 스키마를 정의
- data-storage 계층 : 지명된 AFF 세그먼트들이 실제 파일에 어떻게 저장되는지를 명시

① Disk-representation 계층

. 디스크 이미지에 관련된 모든 정보를 나타내기 위해 사용되는 특정한 세그먼트 이름들을 정의

. AFF 세그먼트는 세그먼트 name, 32-비트 flag, data payload로 구성됨

- name과 data payload는 0 ~ $2^{32}-1$ bytes의 가변 길이이다. 보통 name은 32 bytes 이하이고, data payload는 16MB 이하이다.

. 두 가지 종류의 세그먼트 지원

- 메타데이터 세그먼트 : 디스크 이미지에 관한 정보 유지
- 데이터 세그먼트 : page 라고 불리며, 이미지된 디스크 정보 자체를 유지

. 메타 데이터 세그먼트는 디스크가 액세스되거나 디스크가 이미지된 후에 생성됨.

Segment Name	AFFLIB Symbol	Meaning
Housekeeping Segments		
—	AF_IGNORE	Ignore this segment; zero-length name.
dir	AF_DIRECTORY	AFF directory; should be the last segment.
AFF Segments (pertaining to the entire image)		
pagesize	AF_SEGSIZE	Size (in bytes) of each uncompressed AFF data page is stored in segment "flag" field.
imagesize	AF_IMAGESIZE	Size (in bytes) of the complete image is stored in the segment data area. (8-byte value).
badsectors	AF_BADSECTORS	Number of 512-byte sectors in the image that were marked as "bad."
badflag	AF_BADFLAG	512-byte flag that is used to denote sectors in the image file that could not be read from the media.
blanksectors	AF_BLANKSECTORS	Number of sectors in the image file that are completely blank (i.e., filled with 512 ASCII NULLs.)
AFF Pages (repeated for each data segment)		
page%d	AF_PAGE	The actual data of page %d. Data is compressed if the AF_PAGE_COMPRESSED bit in the flag is set.
page_md5_%d	AF_PAGE_MD5HASH	MD5 of the uncompressed page.
page_md5_sig%d	AF_PAGE_MD5SIG	PKCS7 signature of page %d's MD5.
page_sha1_%d	AF_PAGE_SHA1HASH	SHA-1 of the uncompressed page.
page_sha1_sig%d	AF_PAGE_SHA1SIG	PKCS7 signature of page %d's SHA-1.
Some AFF Segments created by the Advanced Disk Imager		
case_num	AF_CASE_NUM	Case number for EnCase compatibility.
image_gid	AF_IMAGE_GID	A unique 128-bit image identifier.
imaging_commandline	AF_IMAGING_COMMAND_LINE	Complete command used to create image.
imaging_date	AF_IMAGING_DATE	Date and time when imaging was started.
imaging_notes	AF_IMAGING_NOTES	Notes made by the forensic examiner when the imaging was started.
imaging_device	AF_IMAGING_DEVICE	Device used as the source of the image.

[표 8] AFF 1.0 스펙에 정의된 AFF 세그먼트들

. 데이터 세그먼트는 조사되는 디스크로부터 복사된 실제 데이터를 저장

- 모든 데이터 세그먼트들은 동일한 크기여야 하며, 그 크기는 이미지 파일이 생성되고 pagesize라 불리는 세그먼트에 저장될 때 결정된다.
- pagesize는 임의의 값을 가질 수 있지만 보통의 경우 2^{20} (1MB)와 2^{24} (16MB)가 선택된다.

. 세그먼트들은 순차적인 이름 page0,page1, ... page n 으로 주어진다. 여기서 n은 필요한만큼 커진다.

. AFF 파일에서 임의의 512-바이트 섹터 i의 압축되지 않은 세그먼트 내에서 세그먼트(page) 번호와 바이트 오프셋은 다음 식에 의해 결정된다.

$$\text{page number} = \frac{i \times 512}{pagesize}$$

$$\text{offset} = (i \times 512) - (page\# \times pagesize)$$

. AFF 데이터 페이지는 오픈 소스 zlib로 압축되거나 또는 압축되지 않을 수 있다.

- 데이터 페이지의 32-비트 flag 가 페이지가 압축되었는지의 여부를 encode 한다.

. checksum 과 signature 는 이미지 데이터 세그먼트에서 데이터가 획득된 이후에 의도적이거나 사고로 변경되지 않았다는 것을 인증하기 위해 사용될 수 있다.

- AFF 1.0은 해쉬(MD5,SHA-1)와 전자 서명(PKCS #7 전

자서명)의2 가지 인증모드를 지원한다.

- 해쉬와 전자 서명은 전체이미지에 대해 그리고 개별 데이터 세그먼트에 대해 기록될 수 있다.
- signature 는 압축되지 않은 데이터에 계산되기 때문에 디스크 이미지를 획득하여 서명한 후 이미지를 압축하는 것이 가능하다.

② Data-Storage 계층

. data-strage 계층은 실제 AFF 세그먼트를 저장하는 다룸

. 두 가지 저장 포맷을 가짐

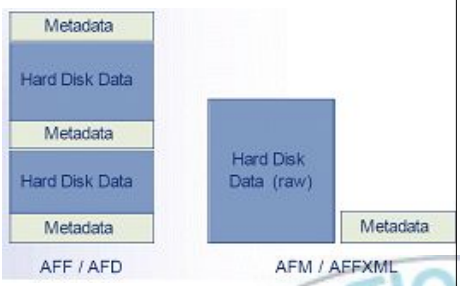
- AFF 1.0 Binary format : 세그먼트들이 하나 이상의 파일에 순차적으로 저장된다는 것을 명시
- AFF 1.0 XML format : XML 포맷으로 정보를 저장

. AFF 1.0 Binary file

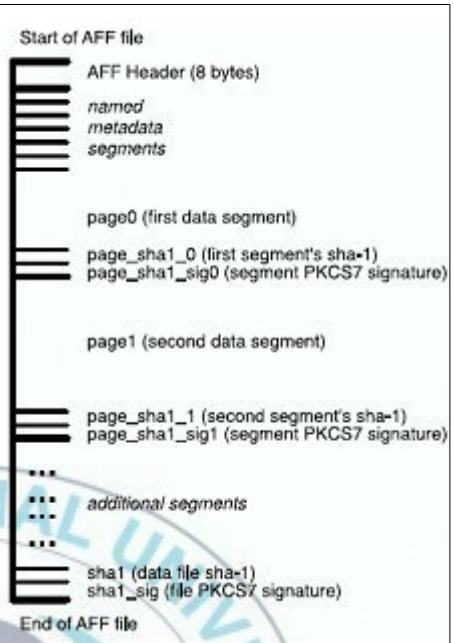
- header 와 0개 이상의 binary AFF 세그먼트로 구성
- 각 binary AFF 세그먼트는 header, 가변 길이 세그먼트 이름, 32-비트 flag, 가변 길이 데이터 영역, 세그먼트 footer 로 구성
- 세그먼트의 길이는 header 와 footer 에 저장

. AFF1.0 XML format

-일련의 XML 객체들로 name/value 세그먼트의 간단한인코딩

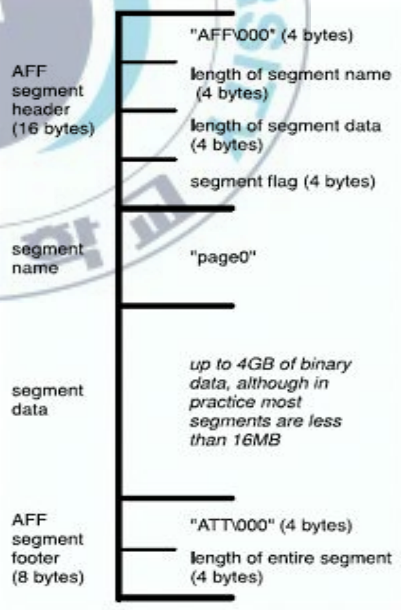


< AFF Storage Formats >



< AFF file schematic >

< AFF Segment Formats >



< AFF segment schematic >

[그림 3] AFF1.0 XML format

III. 대용량 저장 매체 획득 포렌식 툴 파일 포맷 연구

디스크 이미징은 컴퓨터 포렌식에 있어 증거물 획득을 위한 저장 미디어의 내용을 비트 스트림 방식을 통해 파일 형태로 저장하는 일련의 행위를 의미하며 이를 위한 다양한 하드웨어 장비와 소프트웨어 툴들이 알려져 있다. 디스크 이미징에 의해 저장되는 파일은 크게 다음과 같은 두 가지 형태로 분류될 수 있다.

- Raw Data
- Formatted Data

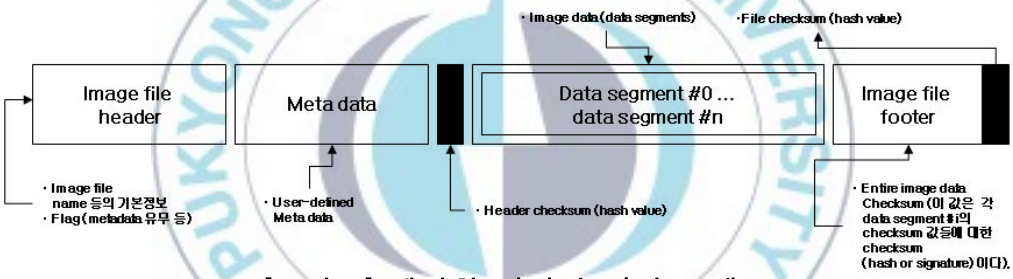
Raw Data" 형식은 순수한 디스크의 내용만을 저장하고 있는 방식으로 유닉스/리눅스 시스템이 제공하는 기본 명령어인 "dd"에 의해 획득될 수 있다. "Formatted Data"는 대부분의 상용 포렌식 툴들 혹은 최근의 이미징 관련 공개 소프트웨어들이 지원하고 있으며 이는 순수 데이터에 다양한 부가정보들을 덧붙이거나 때로는 원본 데이터를 관리 목적으로 임의의 크기로 분할함으로써 얻어지는 형태이다. 가장 잘 알려진 포렌식 툴인 EnCase를 포함하여 GPL 선언에 따라 개발 후 공개된 AFF가 "Formatted Data" 형태의 이미지 파일을 작성한다.

하지만 최근 개인 사용자들에게도 100 기가 바이트 이상의 대용량 하드디스크가 보편화되면서 컴퓨터 포렌식에 있어서도 증거획득과 분석에 대한 새로운 패러다임이 형성되고 있다. 따라서 컴퓨터 포렌식을 위한 디스크 이미징에 있어서도 이러한 고용량 저장 장치의 데이터 획득과 관리가 가능한 이미지 포맷이 요구되고 있다. 또한 조사관의 다양한 기술정보들이 자유롭게 첨부될 수 있도록 확장이 용이한 데이터 구조와 함께 여러

가지 제어 정보를 포함하고 있어 실제적인 증거 획득 과정에서도 안정적으로 사용할 수 있어야 할 것이다.

1. 제안된 이미지 포맷

제안된 이미지 포맷에 의한 개별 이미지 파일은 그림 4과 같이 구성된다. 동일한 증거물로부터 획득된 이미지 파일이 분할되어 저장되었을 경우에도 모든 이미지 파일들이 동일한 구성을 가진다.



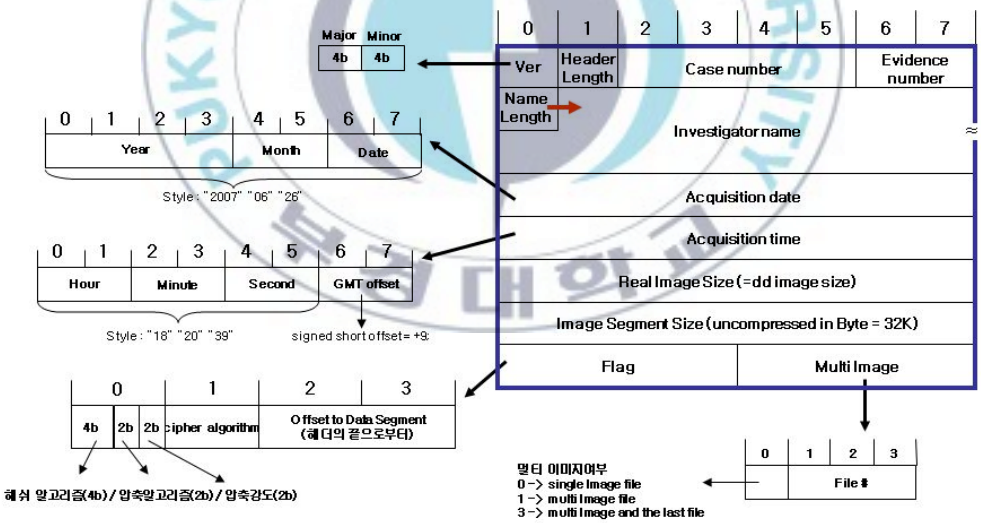
[그림 4] 제안한 이미지 파일 포맷

각 부분의 개별 구성 및 데이터 구조에 대한 설명은 다음과 같다.

2. 이미지 파일 헤더

그림 5는 이미지 파일 헤더의 데이터 구조를 나타낸 도면이다. 그림 2에서 'Ver' 는 1바이트 길이를 가지며 제안한 이미지 포맷의 버

전(Version)번호를 나타낸다. 1바이트 중, 상위 4비트(Bit)는 메이저(Major) 번호를 하위 4비트는 마이너(Minor)번호를 나타낸다. 만약 버전 번호가 0.9일 경우 'Ver'의 비트맵(Bitmap)은 '00001001'이 된다. '헤더길이'는 1바이트 길이를 가지며 파일 헤더의 바이트 길이를 나타낸다. 따라서 파일헤더는 256바이트 이상의 길이를 가질 수 없다. '사건번호'는 4바이트 길이를 가지며 제안한 이미지 포맷으로 획득된 증거물과 관련한 사건을 수치로 표현한 사건번호(Case Number)를 표시한다. '증거번호'는 2바이트 길이를 가지며 '사건번호'에 해당하는 사건과 관련된 증거물들에 부여된 일련의 번호 중 본 이미지 파일에 해당하는 증거번호(Evidence Number)를 나타낸다.



[그림 5] 이미지 파일 헤더 포맷

'이름길이'는 1바이트 길이를 가지며 '조사관이름'의 길이를 나타낸다. '조사관이름'은 0 바이트 이상 191 바이트 이하의 길이

를 가질 수 있으며 제안된 이미지 파일의 획득 혹은 분석을 담당하는 인물의 이름을 기입하는 항목이다. ‘획득날짜’는 8바이트 길이를 가지며 제안한 이미지 파일 포맷에 따라 이미지 파일을 작성한 날짜 정보가 기입되는데 이의 상위 4바이트는 해당 년도(Year)에 해당하는 4문자(Character)가 다음 2바이트에는 달(Month)에 해당되는 2문자가, 마지막 2바이트에는 날(Day)에 해당하는 2문자가 차례대로 기록된다. 일례는 만약 이미지 파일의 작성일이 2007년 7월 5이라면 ‘획득날짜’의 8바이트에는 각각 ‘2’, ‘0’, ‘0’, ‘7’, ‘0’, ‘7’, ‘0’ ,’ 5’ 의 문자가 기록된다. ‘획득시간’은 8바이트 길이를 가지며 제안한 이미지 파일 포맷에 따라 이미지 파일을 작성한 시간 정보가 기입되는데 이중 상위 2바이트에는 시각(Hour)에 해당하는 2문자가, 다음 2바이트에는 분에 해당하는 2문자가, 다음 2바이트에는 초(Second)에 해당하는 2문자가 기입된다. 마지막 2바이트엔 GMT timezone을 나타내는 값이 기록된다. 따라서 대한민국 기준시(GMT timezone 이 +9)로 23시 12분 10초를 표현하기 위해서는 ‘획득시간’의 초기 6바이트에 각각 ‘2’, ‘3’, ‘1’, ‘2’, ‘1’, ‘0’의 문자가, 다음 1바이트에는 ‘+’라는 문자가 기록되며 마지막 1바이트는 비트맵이 0x09가 된다. ‘이미지사이즈’는 8바이트 길이를 가지며 제안한 이미지 파일 포맷을 이용한 이미지 파일 작성의 대상이 되는 증거물의 바이트 크기를 나타낸다. 따라서 저장용량이 2^{64}바이트 이상인 저장 매체 혹은 2^{64}바이트 이상의 크기를 가지는 데이터 파일에 대한 이미지 파일 작성에 대해서는 본 이미지 파일 포맷의 사용이 권고되지 않는다. ‘세그먼트사이즈’는 8바이트 길이를 가지며 데이터 세그먼트에 포함되는 원본 내용의 크기를 나타내는 킬로바이트 단위로 나타낸다.

'Flag'은 4바이트 길이를 가지며 여러 가지 부가적인 제어 정보들이 기입된다. 'Flag'의 초기 4비트는 해쉬 알고리즘을 나타내기 위한 것으로, 기록된 비트맵이 0x0일 경우는 MD5, 0x1일 경우 SHA-1, 0x2일 경우 RSA와 MD5를 이용한 전자서명, 0x3일 경우 RSA와 SHA-1을 이용한 전자서명, 0x4일 경우 DSA와 MD5를 이용한 전자서명, 0x5일 경우 DSA와 SHA-1을 이용한 전자서명을 사용함을 의미한다. 다음 2비트는 압축 알고리즘을 나타내며 비트맵이 0x0일 경우는 무압축(No Compression)을, 0x1일 경우 zlib 사용을 의미한다. 다음 2비트는 압축의 강도를 의미하며 비트맵이 0x0은 zlib 에 의한 압축 레벨(level) 6를 0x1일 경우 압축 레벨 9를 의미한다. 'Flag'의 두 번째 1바이트는 사용된 암호화 알고리즘을 나타내는데 비트맵이 0x00일 경우 암호화를 사용하지 않음을 의미하며 0x01은 3DES, 0x02는 AES(128비트)를 사용하여 암호화함을 의미한다. 'Flag'의 마지막 2바이트는 제안한 이미지 포맷에 의한 이미지 파일의 시작으로부터 첫 번째 세그먼트데이터까지의 거리에 대한 바이트값을 나타낸다. '멀티이미지'는 4바이트 길이를 가지며 하나의 증거물로부터 획득된 데이터가 복수의 이미지 파일로 순차적으로 분할되었을 경우 분할 여부와 전체 분할 파일 집합에서 해당 파일의 위치를 나타내는 정보가 기록되는데 첫 번째 바이트는 0x00, 0x01, 0x03의 비트맵만을 가져야 하며 이는 각각 '단일 이미지 파일', '분할 이미지 파일', '분할 이미지 파일의 끝'을 의미한다.

3. 메타 데이터 (Metadata)

그림 3은 메타데이터의 구조를 도시하고 있다. 그림에서 'ID'는 1바이트 길이를 가지며 메타데이터를 구분하기 위한 식별자(Identification Number)로 사용한다. 따라서 본 논문에서는 최대 256개의 메타데이터가 정의되어 사용될 수 있다. 'Length'는 4바이트 길이를 가지며 메타데이터 전체의 길이에 대한 바이트 값을 나타낸다. 'Data'는 가변 길이를 가지며 메타데이터 별 정의에 따라 다양한 정보를 가질 수 있다. 기본적으로 메타데이터는 복잡하지 않은 포맷과 가변적인 데이터 길이를 사용하기 때문에 사용자가 임의로 새로운 메타데이터 형을 추가할 수 있으며 항상 이미지 파일 내에 존재할 필요가 없다.

[그림 6] 메타 데이터 포맷

표 9는 본 논문에서 미리 예약한 메타데이터의 내용과 ID 번호를 나타내고 있다.

[표 9] 정의 된 메타 데이터

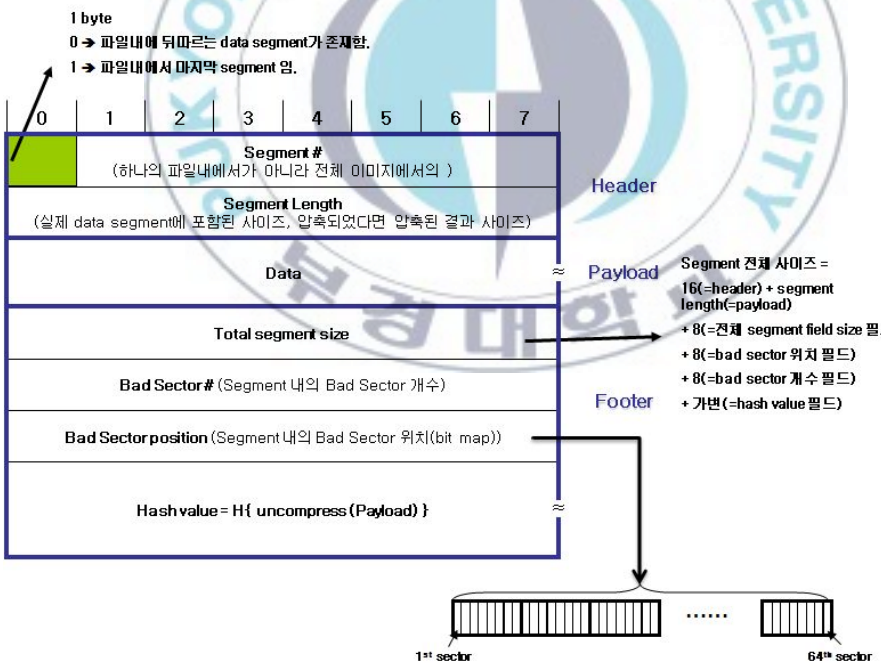
ID	DATA 내용	기타
0x00	디스크 Serial Number	
0x01	디스크 제조사/모델명	
0x02	디스크 드라이브 기술명	
0x03	디스크 용량(크기/섹터수/섹터당 바이트수)	

0x04	운영체제 및 버전 정보	
0x05	전체 이미지 파일에서 bad sector 개수	64bits Integer
0x06	이미지 툴 및 버전 정보 (dd 의 경우는 명령식도 표현)	
0x07	Unique Identifier for this file	
0x08	RSA public key로 암호화된 블록암호화 키	암호화는 Segment의 Payload 부분만 수행
0x09	전체 디스크 이미지를 입력 데이터로 한 해쉬 값	MD5 or SHA1
0x0A	증거 컴퓨터의 제조사	
0x0B	증거 컴퓨터의 모델	
0x0C	증거 컴퓨터의 Serial Number	
0x0D	압수 장소	
0x0E	압수 당시의 증거 컴퓨터 날짜	FileHeader의 획득 시간과 동일한 포맷
0x0F	압수 당시의 증거 컴퓨터 시간	FileHeader의 획득 시간과 동일한 포맷
0xFF	Header 로부터 이 Metadata의 앞부분까지의 해쉬값	Metadata의 끝을 알린다 MD5 or SHA1

4. 세그먼트 데이터 (segment data)

그림 7는 세그먼트데이터의 데이터 구조를 도시하고 있다. 그림 5에서 'segment #'은 8바이트 길이를 가지며 최상위 1바이트가 1일 경우는 제안된 데이터 포맷을 가지는 이미지 파일 내에서 해당 세그먼트데이터가 마지막 세그먼트데이터임을 나타낸다. 나머지 7바이트는 세그먼트데이터의 순서값을 나타낸다. 즉, 32768바이트의 크기를 가지는 하드 디스크를 본 논문에서 제안한 데이터 포맷으로 하나의 이미지 파일을 작성할 경우 그림 2의 '이미지사이즈' 비트맵과 그림 5

의 ‘segment length'의 최상위 1바이트의 비트 맵은 각각 ‘0x0000000000008000’과 ‘0x00’이 된다. 만약 ‘세그먼트사이즈’의 비트맵을 ‘0x0000000000000008’으로 설정했다면 이는 원본을 8킬로바이트(=8192바이트) 단위로 분할하여 세그먼트데이터에 포함한다는 의미가 되므로 전체 세그먼트의 개수는 4(=32768/8192)개가 된다. 따라서 원본의 첫 번째 8킬로바이트가 포함된 세그먼트데이터의 ‘segment #'의 비트맵은 ‘0x0000000000000000’이, 두 번째 8킬로바이트가 포함된 세그먼트데이터의 ‘segment #'의 비트맵은 ‘0x0000000000000001’이 될 것이다.

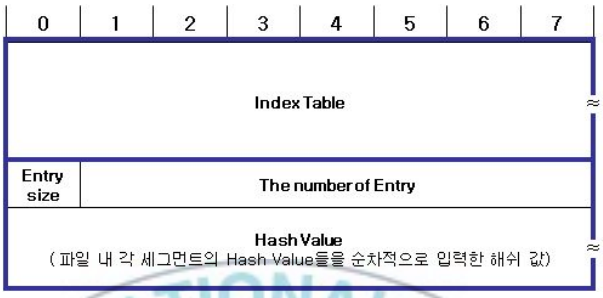


[그림 7] 세그먼트 데이터 포맷

'segment length'는 8바이트 길이를 가지며 그림 7의 'data'의 바이트 크기를 나타낸다. 만약 압축이 이루어졌다면 압축 후의 바이트 크기를 기록한다. 따라서 압축이 이루어지지 않았다면 그림 2의 '세그먼트사이즈' 에 기록된 값에 1024를 곱한 결과값과 'segment length'에 기록된 값은 동일하여야 하며 압축이 이루어졌다면 같거나 작아야만 한다. 'total size'는 8바이트 길이를 가지며 세그먼트데이터의 전체 크기를 나타낸다. 'bad sector #'은 8바이트 길이를 가지며 세그먼트에 포함된 원본 디스크의 내용 중 불량 섹터의 개수를 나타낸다. 섹터의 크기는 일반적인 컴퓨터 분야의 관례적 사용에 따라 512바이트 길이로 가정한다. 'bad sector position'은 그림 2의 '세그먼트사이즈' 의 비트맵값을 10진수로 환산한 값의 2배에 해당하는 길이를 가지며 'data'에 저장되는 압축되지 않은 원본데이터의 각 섹터에 대한 비트맵을 가진다. 즉, '세그먼트사이즈' 의 비트맵이 '0x00000000000008' 이라면 세그먼트데이터에 포함되는 원본의 크기가 8킬로바이트가 되므로 16개가 섹터로 이루어진다. 이런 경우 'bad sector position'은 2바이트 길이를 가지게 되며 각각의 비트가 개별 섹터에 대응되고 불량 섹터의 해당 비트는 1로 세팅된다. 'hash value'은 'data'에 기록되는 압축 전 원본 데이터의 해쉬값 또는 전자서명값이 기록되며 그 크기와 사용하는 알고리즘은 그림 5의 'Flag' 의 최상위 4비트의 비트맵에 따라 길이가 달라진다. 최상위 4비트의 비트맵이 0x0, 0x2, 0x4일 경우 MD5 알고리즘을 사용하며 16바이트(=128비트)길이를 가지고, 비트맵이 0x1, 0x3, 0x5일 경우 SHA-1알고리즘을 사용하며 20바이트(=160비트) 길이를 가진다.

5. 파일 종결부 (file footer)

그림 8은 파일종결부의 데이터 구조를 도시하고 있다.



[그림 8] 파일 종결부 포맷

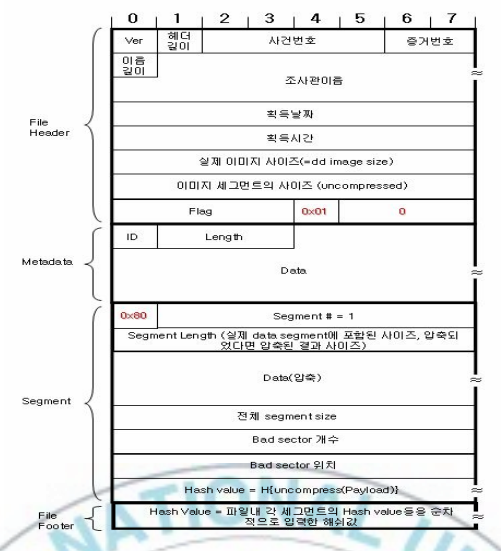
그림에서 'entry size'은 1바이트 길이를 가지며 'index table'에 포함되는 엔트리(Entry)의 기본 단위를 바이트 크기로 나타내며 비트맵이 0x00일 경우 'index table'은 이미지 파일내에 존재하지 않아야 한다. 'table size(the number of Entry)'는 7바이트 길이를 가지며 'index table'에 포함될 엔트리의 개수를 나타낸다. 'index table'은 'entry size'에 기입된 엔트리 크기와 'table size'에 기입된 엔트리 개수의 곱에 해당하는 값만큼의 길이를 가진다. 'index table'에 기입되는 개별 엔트리는 본 논문에서 제안한 데이터 포맷을 사용한 이미지 파일의 시작으로부터 각 세그먼트데이터들의 거리를 바이트 단위로 나타낸 값을 가진다. 그림 8의 설명에서 예를 든 것처럼 만약 32768바이트의 디스크를 8킬로바이트씩 세그먼트데이터에 포함하고 하나의 이미지 파일로 만든다면 모두 4개의 세그먼트데이터가 생기게 된다. 이럴 경우 'table size'의 비트맵은 ‘0x00000000000004’ 이 된다. 'index table'의 경우는 최종적인 이미지 파일의 크기를 기입함이 이상

적이나 이는 인덱스 파일을 작성하는 단계에서 정확하게 예측할 수 없다. 다만 'entry size'의 비트맵을 0x04로 설정할 경우 엔트리가 4바이트 크기를 가지고 이는 4G(=2^{32})바이트의 파일 크기를 표현할 수 있으므로 현실적으로도 충분히 사용할 수 있다. ‘index table'을 이용할 경우 개별 세그먼트데이터로의 접근이 순차적이 아니라 테이블을 이용한 임의 접근(Random Access) 수준으로 이루어 질 수 있다.

'hash value'는 각 세그먼트데이터에 포함된 Hash value를 연접(Concatenation)한 후 해쉬 함수에 적용한 결과 값을 기록한다. 사용되는 해쉬 함수는 'Flag'의 최상위 4비트 값의 비트맵이 0x0일 경우 MD5만 적용하고 그 결과인 16바이트(=128비트)를 기록한다. 0x1일 경우 SHA-1만 적용하고 그 결과인 20바이트(=160비트)를 기록한다. 0x02일 경우 MD5결과를 RSA 비밀키(private key)로 암호화하고 그 결과인 128바이트(=1024비트)를 기록한다. 0x03일 경우 SHA-1결과를 RSA 비밀키로 암호화하고 그 결과인 128바이트(=1024비트)를 기록한다. 0x04일 경우 MD5결과를 DSA 알고리즘으로 암호화하고 그 결과인 128바이트(=1024비트)를 기록한다. 0x04일 경우 SHA-1 결과를 DSA 알고리즘으로 암호화하고 그 결과인 128바이트(=1024비트)를 기록한다.

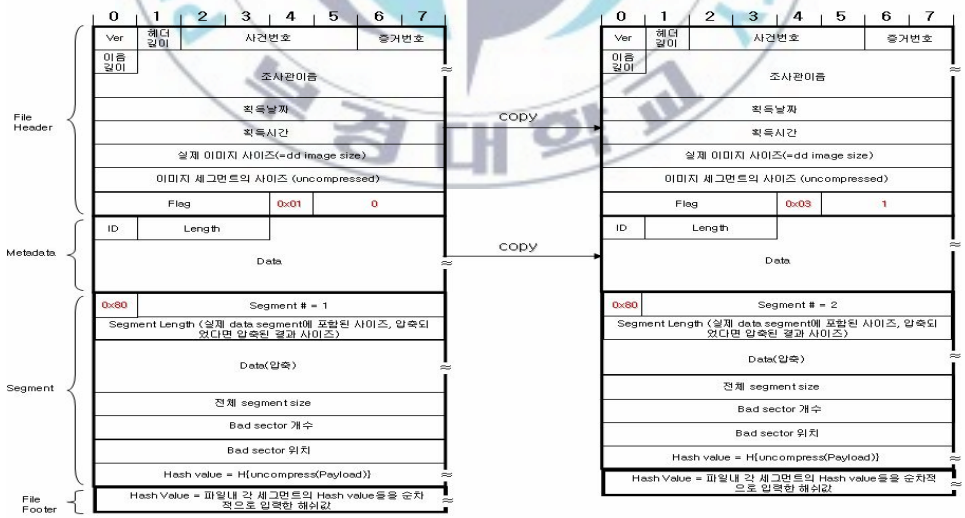
6. 이미지 파일의 전체 구조

이미지파일의 전체 구조를 하나의 그림으로 표시하면 다음과 같다.



[그림9] 이미지 파일의 전체 구조

붉은색 글씨는 segment의 개수에 따라 또는 이미지파일의 개수에 따라 변경될 수 있음을 표시한다. 이미지 파일이 복수개로 저장될 경우 처리는 그림 10과 같다.



[그림 10] 복수파일 처리

IV. 디스크 이미지 교환 포맷 연구

디지털 증거를 획득하기 위해서는 범죄의 증거가 있다고 판단되는 물리적, 혹은 논리 단위의 디스크에서 데이터를 수집하고 수집된 데이터를 근거로 사건을 재연할 수 있는 정보를 추출하게 된다. 이때 수집된 데이터는 일반적으로 이미지 파일 형태로 저장되며, 무결성, 원본성, 진정성 등을 보장하기 위한 수단이 제공되어야 한다.

현재까지 수집된 디스크 이미지에 대한 포맷은 응용 프로그램이나 도구 개발업체에 따라 상이함으로 인해 특정 분석 도구에 의존적인 수사로 제한되어 왔다. 그러나 디지털 포렌식은 사이버 범죄의 특성상 원격지에서 수집한 데이터에 대해 증거 분석을 실행해야 하는 경우가 발생할 수 있으며, 분석된 증거의 신뢰성 입증을 위해 한번 수집한 데이터를 이용하여 다양한 분석이 요구될 수 있다. 그러므로 수집된 데이터의 원활한 교환 및 이기종 도구에서 수집한 데이터를 분석할 수 있도록 하기 위한 공통된 교환포맷이 필요하다.

이에 이장에서는 다양한 디스크 이미지 포맷을 수용할 수 있으며, 더불어 데이터의 무결성, 원본성, 진정성을 보장할 수 있는 수단을 제공하며, 디스크 이미지뿐만 아니라 기타 다른 증거 파일들 저장에도 활용할 수 있다. 이를 통해 응용 프로그램 및 시스템에 독립적인 디지털 증거 분석이 가능하고, 기관간, 국가간 수집된 디지털 증거 파일 교환을 통해 국제 범죄에 대한 수사 공조가 원활해 질 것을 기대할 수 있다.

1. 디스크 이미지 교환 포맷에 대한 요구 사항

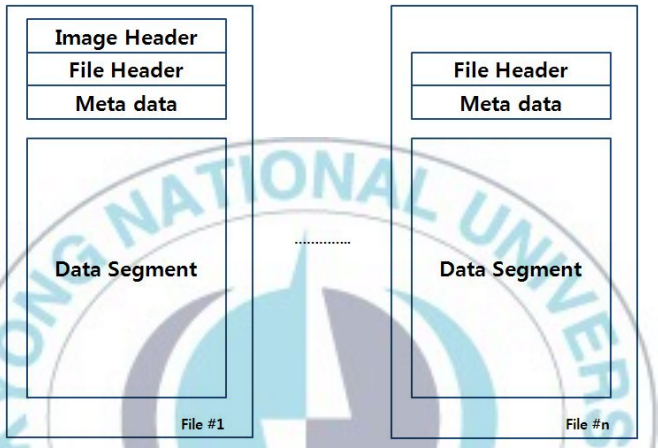
디지털 증거 수사를 위해 국가 간, 기관 간, 도구 간의 이미지 교환을 위한 교환포맷은 아래와 같은 요구사항을 만족해야 한다.

- 이미지 교환에 필요한 정보를 담을 수 있어야 한다.
  : 이미지 교환포맷은 증거 파일 작성과 관련한 정보(작성기관, 작성시간 등)와 증거 파일 내에 포함되어 있는 데이터에 대한 정보 등을 포함해야 한다.
- 기존 상용제품의 이미지 포맷을 수용할 수 있어야 한다.
  : 기존에 많이 쓰이고 있는 디지털 포렌식 제품이 작성하는 이미지 포맷을 담을 수 있어야 한다.
- 무결성 확인 방법을 제공한다.
  : 이미지 파일의 전송 과정에서 변동이 발생한다면 분석된 증거의 신뢰성에 영향을 줄 수 있으므로 무결성을 검증할 수 있는 방법을 제공해야 한다.
- 발송자 확인 방법을 제공한다.
  : 증거의 연계 보관성 확보를 위해 발송자 확인 방법을 제공해야 한다.
- 파일 분할 기능을 제공한다.
  : 네트워크를 통해 전송되거나 작은 용량의 이동식 저장장치를 이용해 교환하는 경우가 발생할 수 있으므로 작은 용량으로 분할하여 저장할 수 있어야 한다.

2. 디스크 이미지 교환 포맷

가. 디스크 이미지 교환 포맷 기본 구조

이미지 교환 포맷의 전체 구조는 그림 11과 같다.



[그림 11] 디스크 이미지 교환 포맷

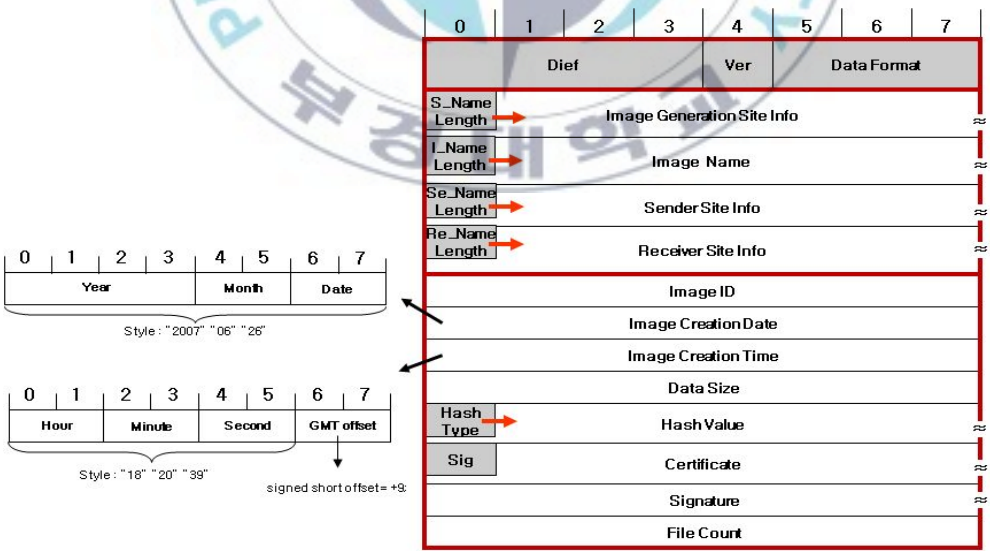
이미지 교환 포맷은 이미지 헤더와 파일 헤더, hash value 그리고 데이터 세그먼트로 구성된다. 이미지 파일헤더는 작성된 디스크이미지 전체에 대한 정보를 저장하고, 파일헤더는 파일에 저장된 데이터 세그먼트의 정보를, hash value는 이미지 파일헤더와 파일헤더 무결성 보장을 위한 값을 가지며, 데이터 세그먼트는 교환하고자 하는 데이터를 포함한다. 교환 대상 데이터는 하드디스크 등에서 수집한 데이터 자체일 수도 있고, 다른 포맷으로 저장된 이미지 파일이 될 수도 있다. 만일 디스크 데이터 자체를 포함한다면 본 포맷을 디스크 이미지 공통 포맷으로 활용할 수 있을 것이다. 또한 다른 포맷으로 저장한 이미지

파일을 본 교환포맷의 데이터로 저장할 수도 있으므로 기존의 이미지 포맷에 대한 호환성을 갖는다 할 수 있겠다. 이때 저장된 데이터의 포맷에 대한 정보는 이미지 헤더에 포함된다.

교환하고자하는 데이터를 하나이상의 이미지로 분할하여 저장할 수도 있으며, 분할하여 디스크 이미지 파일을 생성할 경우 전체 분할 파일의 개수는 이미지 헤더에 저장되고 분할된 파일 중 몇 번째 파일인지에 대한 정보는 파일 헤더에 포함된다. 분할 이미지 파일의 경우 첫 번째 이미지만 이미지 헤더를 가지며, 각 분할 이미지파일은 저장된 데이터 세그먼트가 다르므로 데이터 세그먼트에 대한 정보를 포함하는 파일 헤더의 값은 달라질 수 있다.

나. 디스크 이미지 교환 헤더 포맷

디스크 이미지 교환 포맷 내의 이미지 헤더 구조는 그림 12과 같다.



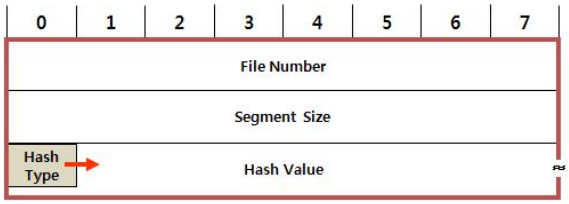
[그림 12] 디스크 이미지 교환 헤더 포맷

그림 12에서 'Dief' 는 4바이트로 dief(Disk Image Exchange Format)의 문자 값을 가지며 이 포맷이 디스크 이미지 교환 포맷이라는 것을 나타낸다. 'Ver'은 1바이트 길이를 가지며 교환 포맷의 버전(Version)번호를 나타낸다. 1바이트 중, 상위 4비트(Bit)는 메이져(Major)번호를 하위 4비트는 마이너(Minor)번호를 나타낸다. 'Data Format' 은 3바이트로 이루어져 있으며 데이터 세그먼트에 저장될 데이터의 형태정보를 나타내며 0일 경우 RAW, 1일 경우 AFF, 2일 경우 FTK, 3일 경우Smart, 4일 경우 EnCase 포맷을 가지며 나머지 값은 Reserved이다. 'S_name Lensth' 는 1바이트 길이를 가지며 'Image Generation Site Info' 의 길이를 나타낸다. 'Image Generation Site Info'는 0바이트 이상 191바이트 이하의 길이를 가지 수 있으며 최초 이미지 작성기관의 이름을 기입하는 항목이다. 'I_name Lensth' 는 1바이트 길이를 가지며 'Image Name' 의 길이를 나타낸다. 'Image Name'은 0바이트 이상 191바이트 이하의 길이를 가지 수 있으며 이미지 작성자의 이름을 나타낸다. 'Se_name Lensth' 는 1바이트 길이를 가지며 'Sender Site Info' 의 길이를 나타낸다. 'Sender Site Info'은 0바이트 이상 191바이트 이하의 길이를 가지 수 있으며 보내는 파일명을 기입하는 항목이다. 'Re_name Lensth' 는 1바이트 길이를 가지며 'Receiver Site Info' 의 길이를 나타낸다. 'Receiver Site Info'은 0바이트 이상 191바이트 이하의 길이를 가지 수 있으며 받는 파일명을 기입하는 항목이다. 'Image ID' 는 이미지 식별번호로써 작성기관별 가지는 고유 번호를 나타낸다. 'Image Creation Date'은 8바이트의 길이를 가지며 이미지파일을 작성한 날짜 정보가 기입되는데 이의 상위 4바이트는 해당 년도(Year)에 해당하는 문자(Character)가 다음 2바이트에는 달(Month)에 해당

되는 2문자가, 마지막 2바이트는 날(Day)에 해당하는 2문자가 차례대로 기록된다. 'Image Creation Time'은 8바이트 길이를 가지며 제안한 이미지 파일 포맷에 따라 이미지 파일을 작성한 시간 정보가 기입되는데 이중 상위 2바이트는 시각(Hour)에 해당하는 2문자가, 다음 2바이트에는 분에 해당하는 2문자가, 다음 2바이트에는 초(Second)에 해당하는 2문자가 기입된다. 마지막 2바이트에는 GMT timezone 을 나타내는 값이 기록된다. 'Data Size'에는 'Data Segment' 에 들어갈 데이터의 크기를 나타낸다. 'Hash Type'은 1바이트 길이를 가지며 MD5는 0으로 SHA-1은 1로 세팅된다. 'Hash Value'는 데이터 세그먼트의 데이터 전체에 대한 해쉬 값으로 'Hash Type'에 따른 값을 가진다. 'Sig'은 1바이트 길이를 가지며 뒤에 전송자의 전자서명 값의 유무를 나타낸다. 0일 경우 전자서명 되지 않았고 1이면 RSA-SHA1(1024)를 사용하였고 나머지 값은 Reserved 이다. 'Public Key'는 1024비트 길이를 가지며 'Sig'이 1로 세팅되어질 때 공개키 값을 가진다. 'Certificate'와 'Signature'는 'Sig'이 0이외의 값일 경우 전송자 또는 전송기관 인증서와 서명 값을 가진다. 'File Count'는 이미지 분할 저장을 위해 전체 이미지 파일 개수를 나타낸다.

다. 파일헤더 구조

디스크 이미지 교환 포맷 내에 파일 헤더는 각 파일별 저장된 데이터 세그먼트에 대한 정보를 포함하는 것으로 파일헤더 포맷은 그림 13와 같다.



[그림 13] File Header 포맷

그림 13에서 'File Number'는 8바이트로 분할된 파일 중 몇 번째 파일인지에 대한 파일 번호를 나타낸다. 'Segment Size'는 실제 저장된 데이터 세그먼트의 크기를 나타낸다. 'Hash Type'은 1바이트 길이를 가지며 MD5는 0으로 SHA-1은 1로 세팅된다. 'Hash Value'는 데이터 세그먼트의 데이터 전체에 대한 해쉬 값으로 'Hash Type'에 따른 값을 가진다.

라. Metadata 구조

메타데이터는 이미지 데이터 교환 시에 전송 조사기관 또는 조사관에 의해 추가적인 조사 정보 또는 증거정보들을 임의로 추가하여 전송할 수 있도록 제공된다. 디스크 이미지 교환 포맷 내에 메타데이터 포맷은 그림 14과 같다.

[그림 14] MetaData 포맷

그림 14에서 'ID'는 1바이트 길이를 가지며 메타데이터를 구분하기 위한 식별자(Identification Number)로 사용하며 최대 256개의 메타데이터가 정의되어 사용될 수 있다. 'Length'는 4바이트 길이를 가지며 메타데이터 전체의 길이에 대한 바이트 값을 나타낸다. 'Data'는 가변 길이를 가지며 메타데이터 별 정의에 따라 다양한 정보를 가질 수 있다. 기본적으로 메타데이터는 복잡하지 않은 포맷과 가변적인 데이터 길이를 사용하기 때문에 사용자가 임의로 새로운 메타데이터 형을 추가할 수 있으며 항상 교환 포맷 내에 존재할 필요는 없다.

표 10은 본 논문에서 미리 예약한 메타데이터의 내용과 ID 번호를 나타내고 있다.

[표 10] 정의 된 교환 포맷 메타 데이터

ID	이름	DATA 내용
0x00	Device Time	원본데이터의 날짜/시간
0x01	Device Type	저장장치의 종류
0x02	Device Model	저장장치의 모델 번호
0x03	Device Serial	저장장치의 일련 번호
0x04	Sector Size	섹터당 바이트 수
0x05	Device Sectors	저장장치의 총 섹터 수
0x06	Sector Start	이미지 작성 시작 섹터 번호
0x07	Sector Count	이미지에 포함된 섹터 개수
0xFF	Hash	Header 로부터 이 Metadata의 앞부분까지의 해쉬 값 Metadata의 끝을 알린다 MD5 or SHA1

V. 증거파일 획득 및 디스크 이미지 교환 툴 구현

이 장에서는 앞장에서 정의한 포맷들을 가지고 동작하게 될 구성 블록들의 상세 설계사항들과 정의하고 설계된 증거파일 획득 및 디스크 이미지 교환 툴을 동작과 성능을 테스트 한 결과를 기술 할 것이다.

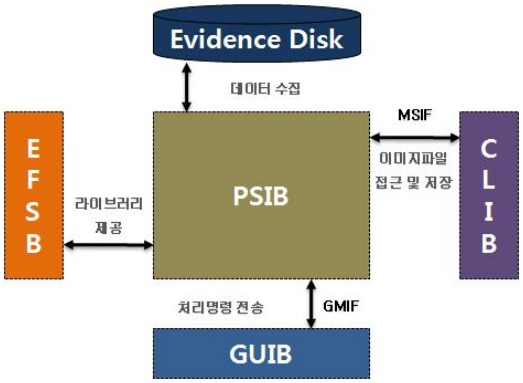
1. 약어 및 용어

- DFS : Digital Forensic System
- HSFS : High Speed Forensic System
- PSIB : Physical Storage Imaging Block
- EFSB : Evidence File Storage Block

2. PSIB의 구성

가. 기능 개요

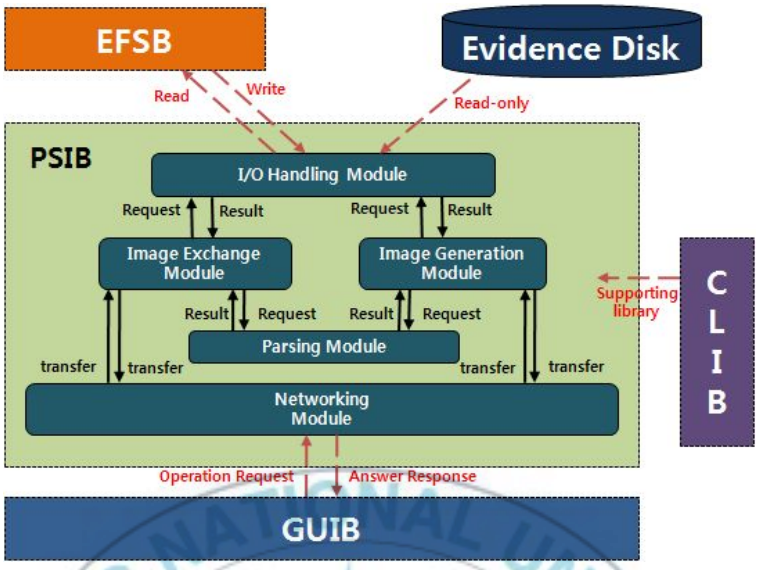
PSIB는 HSFS의 한 구성요소로 리눅스 시스템 상에서 응용프로그램으로 동작한다. PSIB는 GUIB 블록을 통해 전달된 디스크 이미징 요청에 따라 해당 디스크에 저장된 정보들을 bit-by-bit 방식으로 읽은 후, 이를 EFSB를 통해 파일 스트림 방식으로 저장한다. 다음 그림 15는 PSIB의 개괄적 기능 및 이와 연관된 외부 블록과의 연동 내용을 보여준다.



[그림 15] PSIB의 기능 개요

나. PSIB 구조

PSIB는 주 모듈인 Image Generation Module과 Image Exchange Module이 구성하며 이를 보조하기위한 Networking Module, Parsing Module, 그리고 I/O Handling Module로 구성된다. Image Generation Module은 이미징 파일의 포맷에 따라 데이터를 가공하며 Image Exchange Module은 가공된 이미지를 교환포맷으로 가공한다. Parsing Module는 GUIB로부터 전달 된 제어정보(CMSG)를 분석하여 관련변수들에 기록하면 Image Generation Module과 Image Exchange Module이 분석된 제어 내용을 이용하여 이미지 가공 작업 및 교환 포맷 작업을 수행한다. 이때 요구되는 해쉬/압축 암호화등의 작업은 CLIB에서 제공하는 관련 인터페이스 함수들을 이용한다. I/O Handling Module는 PSIB가 EFSB 또는 증거물 디스크(Evidence disk)와의 파일 입출력을 가능하게 한다. 다음 그림16 는 PSIB 내부모듈 상호간 및 타 블록과의 연동을 나타낸다.

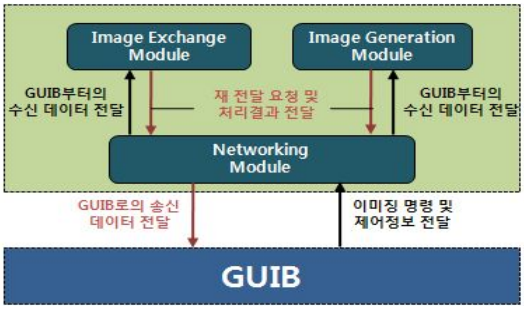


[그림 16] PSIB 내부 모듈간 및 타 블록과의 연동

다. PSIB를 구성하는 개별 블록

(1) Networking Module

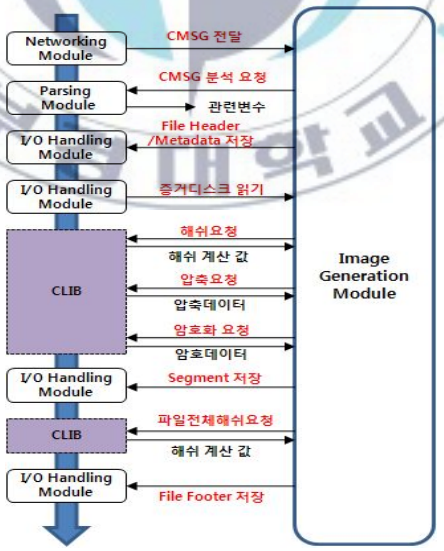
Networking Module은 GUIB로부터 이미징 처리 관련 정보(CMSG)를 내부 버퍼에 저장하여 Image Generation Module에 전달하며, 이 과정에서 에러 혹은 파괴된 정보가 검출 시 Image Generation Module 이 요구하는 재전송 요청과 이미징 처리 완료시의 결과를 GUIB에 전송한다. 이의 구현을 위해 Linux에서 제공되는 Socket Layer 인터페이스를 사용하며, GUIB에 대한 TCP Server로 동작하도록 한다. 그림 17은 이러한 과정을 도시한다.



[그림 17] Networking Module의 동작 과정

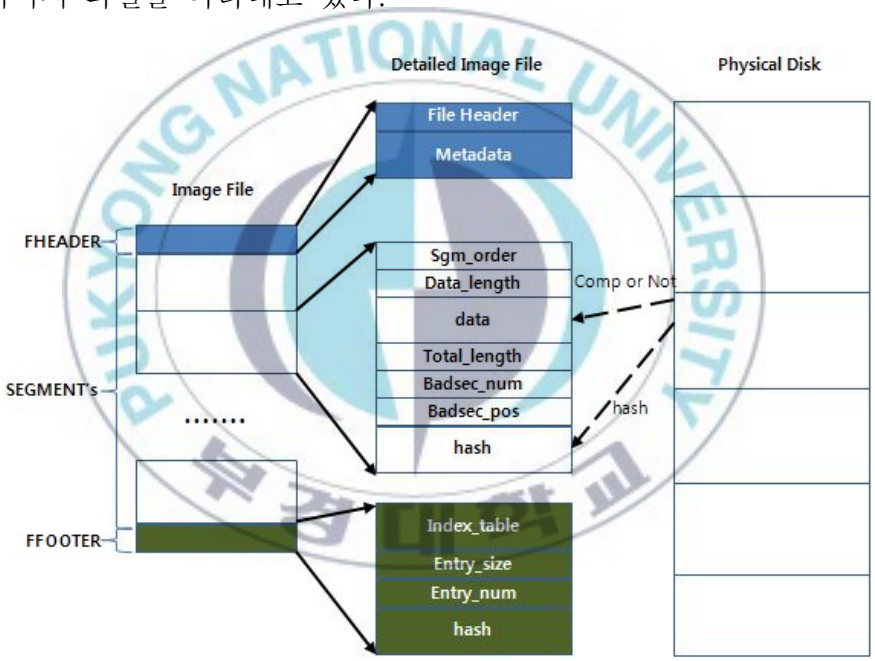
그림 17은 GUIB로부터 GMIF를 통해 PSIB의 Networking Modul로 전달되는 XDR[6]양식의 메시지 데이터(=CMSG) 중, 디스크로부터 포맷된 이미지 파일의 생성을 요청하는 메시지와 생성된 이미지를 교환포맷으로 처리하는 구성을 보여주고 있다.

(2) Image Generation module



[그림 18] 디스크 이미지로부터 포맷된 이미지 파일을 생성하기 위한 프로시져 순서

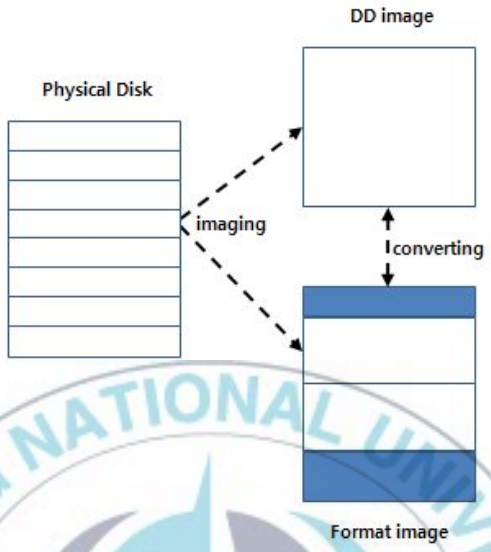
Image Generation Module은 PSIB의 핵심 모듈중 하나로써 Networking Module 로부터 전달받은 데이터(CMSG)를 이용하여 FHEADER 구조체에 값을 저장한 후 해당 디스크에 대한 이미징 작업을 수행한다. 그림 18은 하나의 이미지 파일을 생성하는 과정에서 Image Generation Module과 주변 모듈과의 연동을 시간순서에 따라 표시한 것이다. 그림 19은 그림18과 같은 과정에 의해 생성된 이미지 파일을 나타내고 있다.



[그림 19] Image Generation Module에 의한 이미지파일의 포맷

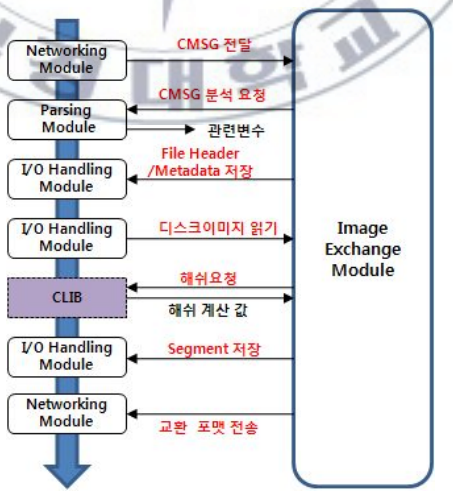
Image Generation Module 은 증거 디스크로부터 그림 20과같이 이미지 파일형태로의 변환기능 외에도 증거디스크의 순수한 이미지 정보만을 기록할 수도 있으며 반대로 생성된 이미지 파일에 대해

서도 상호 변환(포맷이미지->순수이미지, 순수이미지->포맷이미지) 기능도 제공한다.



[그림 20] Image Generation Module에 의한 이미지 생성 및 변환

(3) Image Exchange module

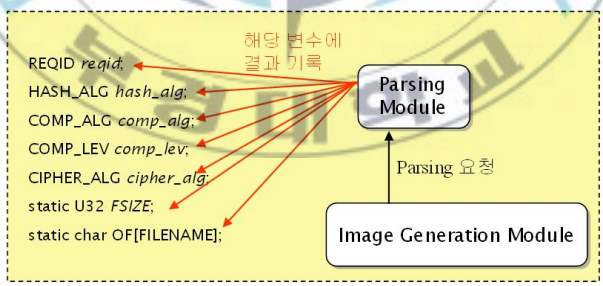


[그림 21] 이미징된 이미지 파일을 교환포맷으로 변환하는 프로시져 순서

Image Exchange Module은 PSIB의 핵심 모듈중 하나로써 Networking Module 로부터 전달받은 데이터(CMSG)를 이용하여 각 해당 구조체에 값을 저장한 후 해당 이미징된 이미지에 대하여 교환 포맷 작업을 수행한다. 그림 21는 이미지 파일을 변환 과정에서 Image Exchange Module과 주변 모듈과의 연동을 시간순서에 따라 표시한 것이다.

(4) Parsing Module

Parsing Module은 Networking Module를 통해 GUIB로부터 전달된 데이터 스트림을 저장한 CMSG 버퍼를 분석하여 그림 22과 같이 해당변수에 그 값을 기록한다. 이 값들은 Image Generation Module에서 XDR_Image Generation Module 관련 자료구조의 내용을 구성하고 다시 Image Generation Module 관련 자료구조로 형 변환을 하여 이미지파일을 생성하는데 이용된다.

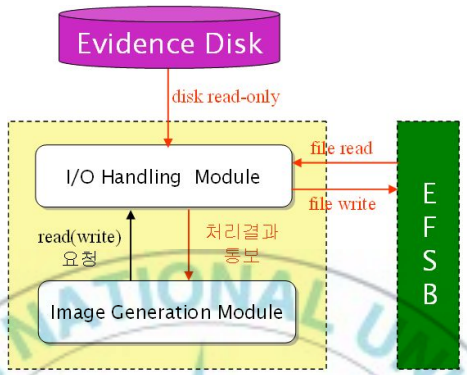


[그림 22] Image Generation Module 와 Parsing Module의 연동내용

(5) I/O Handling Module

I/O Handling Module은 Image Generation Module 이 증거물

(Evidence Disk)로부터 데이터를 읽을 수 있도록 하며 이로부터 작성된 데이터를 EFSB에 전달한다. 그림 23은 I/O Handling Module의 동작과 관련된 타 기능 블록/모듈과의 연동 내용을 보여준다.



[그림 23] Image Generation Module 과 I/O Handling Module의 연동 내용

3. 구현 툴 테스트

가. 정의한 이미지 포맷으로 이미징

(1) TEST 설정

. 디스크는 용량이 크므로 적은 용량의 파일을 이용한 테스트

. source => usb(1G 1031798272byte)

. destination => 0 (임의의 파일명으로 출력하게 설정)

(2) TEST 개요

-임의의 test용 서버를 위의 정보를 바탕으로 동작시킨다.

- dd를 실행(받은 정보에 의해 이미지)

. 파일을 분할하여 이미지 한다.

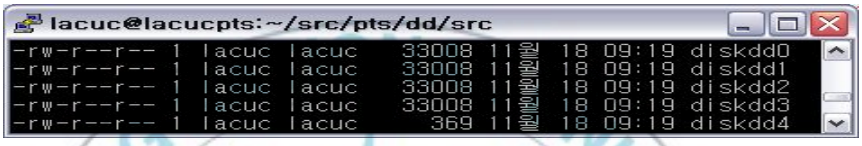
. 하나의 파일로 이미지 한다.

- 이미지 파일생성확인

- readmanager(test용)를 사용하여 이미지에 내용을 확인.

(3) TEST 결과

- 이미지 분할 -



-하나로 이미징 한 결과 -

[그림 24] 정의한 이미지 포맷으로 이미징 결과

증거 디스크에를 이미징 하기위해 디스크 전체 이미징과 디스크를 이미지분할 명령을 주어 각각 수행하였을 때 각 이미징 사이즈로 보아 이상 없이 정의한 포맷대로 이미 징이 이루어졌음 을 확인할 수 있다. 내부 값이 올바르게 이미징 되었는지는 이후 계속 되는 테스트와 해쉬 값을 이용하여 크기 뿐 아니라 내부 값 역시 올바르게 이미징이 이루어졌는지 확인 하도록 한다.

나. RAW Data로 이미징

(1) TEST 설정

. 디스크는 용량이 크므로 적은 용량의 파일을 이용한 테스트

. source => usb(1G 1031798272byte)

. destination => 0 (임의의 파일명으로 출력하게 설정)

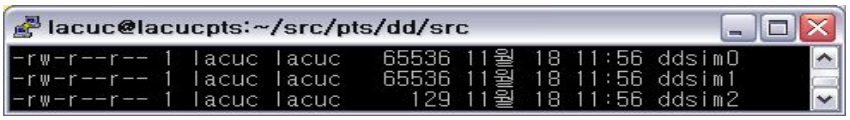
(2) TEST 개요

- dd서버 실행 (받은 정보에 의해 이미지)
  . 파일을 분할하여 이미지 한다.
  . 하나의 파일로 이미지 한다.
- 클라이언트 103으로 실행.
- 이미지 파일생성확인.
- vi편집기를 사용하여 이미지에 내용을 확인.
- 순수 데이터 복사 값임으로 md5sum명령으로 해쉬 값을 비교 한다.

(3) TEST 결과

- 하나의 이미지 -

- 이미지 분할 -



[그림 25] RAW Data로 이미징 결과

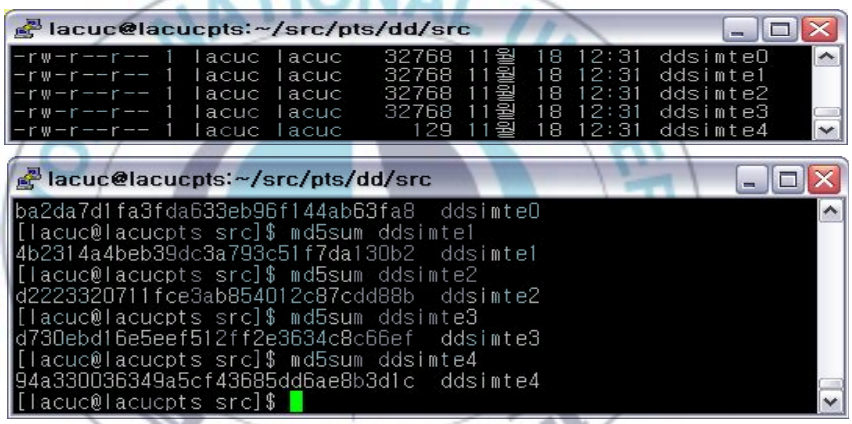
순수 DD이미지로 실행했을 때 싸이즈 비교 시 정확하게 분할되어 짐을 확인할 수 있다.

(4) HASH VALUE 검증

– 하나의 이미지 –

– 이미지 분할 –

(해쉬 값의 비교를 위해 32k단위로 다시파일 분할)



[그림 26] HASH VALUE 검증 결과

간단하게 해쉬 값을 비교해 볼 수 있는 md5sum이란 명령으로 원본 파일과 이미지 파일과의 해쉬 값 비교 및 앞선 이미지포맷으로 이미지를 했을 때 각 세그먼트에 따른 해쉬 값과 32k로 분할한파일의 해쉬 값을 비교했을 때 동일한 결과를 얻었다. 따라서 원본의 이미지기능과 툴의 해쉬 기능이 정확히 한다는 결론을 얻을 수 있다.

다. 이미지 포맷 데이터를 DD이미지로 변환

(1) TEST 설정

. TEST[1]에서 만든 이미지를 이용하여 DD이미지로 변환을 테스트

. source => diskdd0 (TEST[1]에서 만들어진 파일)

. destination => fodd (임의의 파일명으로 출력하게 설정)

(2) TEST 개요

- 임의의 test용 서버를 위의 정보를 바탕으로 동작시킨다.
- dd를 실행(받은 정보에 의해 이미지)
  . 선행 TEST에서 만들어진 이미지를 이용하여 실행
- 이미지 파일생성확인
- vi 편집기를 이용하여 이미지에 내용을 확인.
- md5sum명령으로 변환된 이미지의 해쉬 값을 비교한다.

(3) TEST 결과

- 실행결과 -

- 결과파일 해쉬 확인 -

[그림 27] 이미지 포맷 데이터를 정의 포맷으로 변환

현재 이상이 없는 일반적인 이미지일 경우 변형시키는데 이상이 없으며 해쉬 값 TEST에 의해 변형된 데이터와 원본데이터와 차이가 없음을 확인할 수 있다.

라. DD이미지를 정의 포맷으로 변형

(1) TEST 설정

. TEST에서 복구한 파일을 다시 이미지 포맷으로 바꿈

. source => fodd (32768 byte)

. destination => reim

(2) TEST 개요

- 임의의 test용 서버를 위의 정보를 바탕으로 동작시킨다.
- dd를 실행(받은 정보에 의해 이미지)
  . 선행 TEST[3]에서 만들어진 이미지를 이용하여 실행
- 이미지 파일생성확인
- readmanager(test용)를 사용하여 이미지에 내용을 확인.
- md5sum명령으로 변환된 이미지와 TEST[1]의 해쉬 값을 비교한다.

(3) TEST 결과 및 HASH 검증

- 실행결과 -

- 결과파일 해쉬 확인 -

[그림 28] DD 이미지를 정의 포맷으로 변환

이미지 포맷으로 이미지 한 것을 다시 DD이미지로 그것을 다시 이미지포맷형태로 하나의 파일을 가지고 테스트한 결과 해쉬에 의해 서로 동일한 값들을 가지므로 변형에 있어서도 문제가 없음을 볼 수 있다.

마. 교환이미지로 변환

[그림 29] 정의한 교환 포맷으로 변환

앞선 이미징 TEST에서 만들어진 이미지 파일들을 정의한 교환 포맷으로 변환과정을 수행하였다. 생성된 파일크기와 파일명으로 파일 생성에 문제가 없음을 확인하였고 다른 컴퓨터로 증거 전송 시 파일의 무결성을 검증하기위해 생성된 파일의 해쉬 값을 확인하였다.

바. 교환이미지 전송

[그림 30] 전송받은 교환 포맷

외부 시스템에서 교환포맷으로 변환된 증거 이미지를 전송받아 받은 이미지의 무결성을 검증하기위해 해쉬 값을 이용한다. [그림 30]과 [그림29]의 해쉬 결과 값을 보면 알 수 있듯이 전송 기능과

이미지 복원기능에도 문제가 없음을 확인할 수 있다.

사. TEST 결론

앞선 TEST에 의해 정의된 포맷과 구현되어진 툴이 이상 없이 이미징과 변환 작업을 수행하며 와 HASH검증에 의해 수행한 결과 값이 정확히 이미징 되고 교환됨을 확인할 수 있었다.

VI. 결론

유비쿼터스 컴퓨팅 환경의 조성으로 디지털 포렌식이 수사에서 차지하는 비중이 증가하고 있다. 수사기관은 디지털 포렌식에 관한 역량을 강화시키기 위한 노력을 기울여야한다. 우리나라 정보화 환경은 세계적으로도 최고 수준이지만, 디지털 포렌식의 기술적, 절차적 수준은 그에 미치지 못하고 있다. 따라서 사이버 분야의 국제 경쟁력을 지속적으로 유지하기 위해서는 정보화 환경을 수호하는 노력이 뒷받침되어야한다. 디지털 포렌식의 효율성 증대방안에 대한 많은 논의가 필요한 이유도 여기에 있다.

컴퓨터 포렌식에서 증거물 획득을 위한 저장 미디어의 내용을 비트 스트림 방식을 통해 파일 형태로 저장하는 일련의 행위를 디스크 이미징이라고 하고 이를 위한 다양한 하드웨어장비와 소프트 웨어 툴들이 알려져 있다. 본 연구는 이러한 디스크 이미징을 위한 데이터 획득 툴에 관한 것이다. 대용량 저장 장치의 데이터들에 대한 획득 관리가 가능한 이미지 포맷과 교환 포맷을 정의하여 정의된 이미지 파일 포맷을 지원하는 데이터 획득 포렌식 툴을 구현하고 테스트하였다. 본 논문에서 제안하고 구현된 툴은 이러한 장점을 가진다.

(a) 제안된 이미지 포맷은 최근 저장 매체의 대용량화를 고려하였다.
(b) 인증/압축/암호에 대한 사용자의 설정 항목을 제공하여 다양한 조사환경에서도 충분히 사용할 수 있도록 설계되었다.
(c) 이미지 파일은 파일헤더, 세그먼트데이터, 파일 종결부의 정규 데이터 구조 외에 내부 데이터의 종류나 형식에 구애를 받지 않는 메타데이터를 사용함으로 가변적인 추가정보의 입력이 자유롭다.

(d) 기타 이미지 파일 포맷과는 달리 이미지 파일자체의 보안을 고려하여 암호화 알고리즘의 설정을 파일헤더에 명시할 수 있도록 하였다.

(e) 개별 해당 파일의 시작으로부터 이미지 파일내의 세그먼트데이터들이 가지는 상대적 거리 위치들을 파일종결부내 인덱스 테이블 형태로 기술함으로써 각 세그먼트데이터의 고속 이동이 가능하다.

(f) 각 세그먼트데이터에는 해당 세그먼트 내부의 불량 섹터들의 위치를 비트맵으로 표현하여 불량섹터를 손쉽게 판별할 수 있다.

(g) 다른 포렌식 툴을 사용한 증거 이미지일지라도 툴의 교환기능을 이용해 증거력을 잃지 않고 타 기관과 증거 교환이 가능하다.

이와 같이, 본 논문에서 제안하고 구현한 디지털 포렌식 툴은 기존 포렌식 툴들의 포맷이 가진 장점들을 가지며 또한 새롭게 디지털 이미지 교환 기능을 추가함으로써 포렌식 수사에 있어서 간편하고, 효율적이며 확장성을 제공한다.

VII. 참고 문헌

[1] Shinder D., "Scene of the cybercrime: computer forensics handbook," Syngress Press, Rockland, MA 2002.

[2] S. Garfinkel, D.Malan, K. Dubec, C.Stevens and C. Pham, "Advanced Forensic Format: An Open, Extensible Format For Disk Imaging", in Advances in Digital Forensics: IFIP International Conference on Digital Forensics, National Center for Forensic Science, Orlando, Florida, February 13-16 2005

[3] http:// www.afflib.org.

[4] L. M. Liebrock, et al., " A Preliminary Design for digital forensics analysis of terabyte size data sets," ACM symposium on Applied computing ,pp. 190-191, 2007.

[5] 이상수 외 4명, "대용량 저장 매체를 고려한 디스크 이미지 포맷" , 제1회 안티 포렌식 대응기술 워크샵, pp.67-71,2007.

[6] M.Eisler,"XDR: External Data Representation Standard", IETF RFC 4506, May 2006

[7] Computer Forensics Tool Testing(CFTT) Project, http://www.cftt.nist.gov

[8] Guidance Software, EnCase, http://www.guidancesoftware.com

[9] AccessData, Forensic Toolkit, http://www.accessdata.com

[10] Apple Developer Connection, dd BSD General Commands Manual (developer.apple.com/ documentation/ Darwin/ Reference/ Manpages/man1/dd.1.html).

[11] Armor Forensics, SafeBack (www.forensics-intl.com/safebak.html).

[12] ASR Data Acquisition and Analysis, Expert Witness Compression Format Speci_cation (www.asrdata.com/ SMART/ whitepaper.html), April 7, 2002

[13] ASR Data Acquisition and Analysis, SMART (www.asrdata.com/SMART).

[14] B. Carrier, The Sleth Kit & Autopsy : Forensic Tools for Linux and other Unixes (www.sleuthkit.org), 2005.

[15] Internal Revenue Service, ILook v8 { Computer Forensic Application, IRS Criminal Investigation Division {Electronic Crimes, Washington, DC (www.ilook -forensics.org/homepage.html).

[16] PyFlag, Advanced Open Standard Forensics Format (pyag.sourceforge.net/ Documentation /articles /forensic format.html).

[17] Technology pathways, ProDiscover Image File Format (v.1.3) (www.techpathways.com/ uploads/ ProDiscoverImageFileFormatv4.pdf)

감사의 글

이렇게 2년 동안의 학업에 마침표를 찍는 감사의 글을 쓰면서 감회가 새롭습니다. 더 높은 곳을 향하기위해 대학원을 진학하며 부족한 실력으로 힘들게 노력했던 것들 또 이속에 속해있으면서 한편으로는 많은 꿈을 꾸었고 또 나름에 목표에 충실하기위해 열심히 노력해왔던 지난 2년이 돌이켜보면 많이 아쉽기도 하고 한편 참 보람 있었던 것 같습니다. 많이 부족하고 힘겨워하던 제가 이렇게 논문을 완성할 수 있었던 것은 많은 분들의 관심과 보살핌이 있었기에 가능했던 것 같습니다. 진심으로 감사드립니다.

많이 부족한 저를 2년 동안 지도해주시고 보살펴주신 신상욱 교수님께 먼저 진심으로 너무나 감사드립니다. 그리고 이 논문이 완성되기까지 아낌없는 조언과 지적을 해주시고 가르쳐주신 이경현 교수님과 송하주 교수님 부족함을 채울 수 있도록 많은 것을 배울 수 있도록 정성껏 가르쳐주신 조성진 교수님과 김영봉 교수님 여러 교수님들께 진심으로 감사드립니다.

그리고 2년 동안 연구실에서 많은 조언과 걱정을 해주시며 살갑게 챙겨주신 태훈선배, 친누나같이 연구실에 잘 적응할 수 있도록 챙겨준 수완 선배, 친근하게 다가오셔서 앞으로 공부 방향을 재시해주시던 진홍선배, 항상 좋은 미소로 기분 좋게 해주시는 재성선배, 2년 동안 동고동락하며 때론 나 때문에 많이 고생한 동기 도희, 많이 챙겨주지도 못하고 오히려 날 챙기는 고마운 후배 태림이, 이렇게 2년 동안 연구실 생활 잘하고 보람 있게 해준 연구실사람들 모두에게 감사드립니다.

그리고 날 많이 챙겨주고 열대어 스승이며 다재다능한 석철 선배, 때론 엉뚱하지만 착하고 멋진 동기 원영이, 재미있고 멋진 성현이, 그 외 주위 연구실 사람들 모두 제가 학교에 잘 적응하고 열심히 할 수 있도록 도와주셔서 감사합니다.

마지막으로 묵묵히 나의 곁에서 걱정해주고 응원해주며 언제까지나 함께할 4인용 부광이, 은석이, 형모 내 친구들아 고맙다. 그리고 착하고 날 많이 아껴주는 내 동생 태현이, 어려서부터 키워주신 할머니, 언제나 아낌없는 지원과 걱정을 해주시고 이렇게 학업을 완성할 수 있도록 묵묵히 지켜봐주시고 든든한 힘이 되어 주신 너무나 사랑하는 부모님께 너무나도 고맙고 감사드립니다. 진심으로 모두모두 감사드립니다. 언제나 더 나은 제가 되도록 노력하고 이루도록 하겠습니다.