Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.4, Aug. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.4.911

암호학적 관점에서의 EWF 파일 이미징 효율성 개선 방안 연구*

신 용 학,[1†] 김 도 원,[1] 이 창 훈,[2] 김 종 성[1,3‡]
[1]국민대학교 금융정보보안학과, [2]서울과학기술대학교 컴퓨터공학과, [3]국민대학교 수학과

Improving the Efficiency of the EWF-file Imaging Time from a Cryptographic Perspective*

Yonghak Shin,[1†] Dowon Kim,[1] Changhoon Lee,[2] Jongsung Kim[1,3‡]
[1]Dept. of Financial Information Security, Kookmin University,
[2]Dept. of Computer Science and Engineering, SEOULTECH,
[3]Dept. of Mathematics, Kookmin University

요 약

과거에 비해 현재의 디스크 저장 공간은 비약적으로 증가하고 있으며, 네트워크상에서도 이전과 비교할 수 없는 수많은 데이터들이 처리되고 있다. 이러한 데이터의 대용량화는 앞으로도 계속 될 추세이지만, 그에 비해 포렌식 관점에서 데이터를 이미징 하는 시간을 개선시키기 위한 연구는 부족한 상황이다. 본 논문은 데이터 이미징 시간을 개선시키기 위한 방안으로 전체 이미징 소요시간 중 해시함수에 대한 소요시간을 암호학적 관점에서 살펴보고, 이를 토대로 EWF 파일 이미징 과정에 대한 효율성 개선 방안을 제안한다.

ABSTRACT

Compared to the past, the current disk storages have dramatically increased and extremely many data are transferred on the network everyday. In spite of the anticipation that such development will be continued, there have been lack of studies for improving the data-imaging time in terms of the digital forensics. In this paper, we firstly investigate the time due to hash functions during the data Imaging and secondly propose a method for improving the efficiency of the EWF-File imaging time from a cryptographic perspective

Keywords: Digital Forensic, EnCase, EWF, Image, Hash Functions, SHA-1, MD5

I. 서 론

디지털 포렌식은 수사 감정 등 법·과학 분야로 수사기관을 중심으로 판례에 따라 기술적인 절차와 법, 범죄의 범위, 법의 요구 방향이 정의되었으며, 디지털 포렌식에 대한 요구사항도 다양해졌다. 디지털 포렌식은 일련의 수사 과정을 수행하는 동안 디지털 증거의 무결성 입증 체계를 제공해야 한다. 이는 수사기관의 업무 효율성뿐만 아니라 압수수색을 당하는 피의자의 프라이버시 보호와 사생활 침해를 최소화할 수 있는 방안이기 때문에 중요하다.

일반적으로 디지털 증거에 대한 무결성 입증은 원데이터와 이미지 데이터의 해시값 비교로 이루어진다. 계산한 두 해시값이 동일하다면, 이미지 데이터

Received(04. 06. 2016), Modified(06. 10. 2016), Accepted(06. 13. 2016).
* 이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2013R1A1A2059864).
† 주저자, sinyoohag@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

는 원 데이터와 동일한 것으로 간주하며 디지털 증거에 대한 무결성을 제공한다.

또한 데이터의 대용량화로 인해 디스크 이미징 시간은 나날이 증가하고 있다. 하지만, 데이터의 대용량화의 속도에 비해, 디지털 포렌식 관점에서 데이터의 이미징 속도 개선에 대한 연구는 미흡한 실정이다. 2016년 현재 대부분의 PC는 SATA 방식의 HDD 혹은 SSD를 저장매체로 사용하고 있으며 S/W 기반의 저장매체는 최대 초당 3GB/s 이하의 속도를, H/W 기반의 저장매체는 초당 7GB/s의 속도를 낸다. 이는 저장매체의 노화 상태의 따라 이미징 속도는 천차만별이지만 현재의 이미징 기법은 소프트웨어·하드웨어 모두 한계에 도달한 상태이다.

본 논문에서는 암호학적 관점에서 이미징 효율성 개선방법을 제안한다. 디지털 포렌식 절차 중 디지털 증거물 획득에서 전체 이미징 시간에 대한 해시시간의 비중이나 Delay 의존도를 먼저 파악하며 본 논문에서는 전체 이미징 시간 중 해시시간을 분석하고, 암호학적 관점에서의 속도 향상 방안을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 배경을 설명하고, 3장에서는 관련연구를 소개한다. 4장에서는 EnCase에서 사용되는 EWF 구조에 대해 설명한다. 5장에서는 암호학적 관점에서 효율성이 개선된 EWF 파일 이미징 방식에 대해 제안하며 6장에서는 제안 방식에 대한 실험 결과를 보인다. 끝으로 7장에서는 결론을 맺는다.

II. 배 경

디지털 증거에 대한 동일성·무결성의 원칙은 디지털 데이터의 휘발성·가연성 등의 특징 때문에 물리적인 증거보다 더욱 중요하다. 대법원은 "디지털 저장매체 원본을 대신하여 저장매체에 저장된 자료를 카피 혹은 이미징한 매체로부터 출력한 문건의 경우에는 디지털 저장매체 원본과 카피 또는 이미징한 매체 사이에 자료의 동일성도 인정되어야 한다.[2]에서 디지털 증거의 동일성·무결성의 원칙을 확인하였다. 또한 대법원은 소위 왕재산 간첩 사건에서 "출력 문건과 정보저장매체에 저장된 자료가 동일하고 정보저장매체 원본에 대한 압수, 봉인, 봉인해제, 하드카피 또는 이미징 등 일련의 절차에 참여한 수사관이나 전문가 등의 증언에 의해 정보저장매체 원본과 하드카피 또는 이미징한 매체 사이의 해시값이 동일하다거나 정보저장매체 원본을 최초 압수부터 밀봉되어 증거 제출까지 전혀 변경되지 않았다는 등의 사정을 증명하는 방법 또는 법원이 그 원본에 저장된 자료와 증거로 제출된 출력 문건을 대조하는 방법 등으로도 그와 같은 무결성·동일성을 인정할 수 있다.[3]고 판결하였다. 이를 통하여 디지털 증거의 동일성· 무결성을 보장하는 구체적인 방법을 언급하고 있다.

디지털 포렌식에서 원본의 무결성을 보이기 위하여 해시함수를 사용한다. 해시함수는 가변인 입력의 메시지에 대해 고정된 길이의 출력을 갖는 압축함수이다. 해시함수로는 주로 MD5와 SHA-1을 사용한다(각 해시함수에 대한 스펙은 Table. 1을 참조한다).

데이터 무결성 검증 시 사용되는 순환 검사 방법은 데이터 전송 시 데이터에 오류가 있는지를 확인하기 위한 체크 값을 결정하는 방식을 말한다. 데이터를 전송하기 전에 CRC 값을 계산하여 데이터에 붙여 전송하고, 받은 데이터의 값으로 CRC 값을 계산하여 두 값을 비교하여, 두 값이 다르면 데이터 전송 과정에서 오류가 발생했다는 것을 알 수 있다[1].

디스크를 이미징 과정을 마친 결과인 이미지 파일은 유닉스의 dd 명령에 기초를 둔 고유한 이미징 방식을 사용한다. dd 명령어를 이용하면 외견상 간단히 파일을 복사하는 것처럼 하드디스크를 다른 하드디스크에 복사 가능하다. 만들어진 복사본은 원본 디스크에서 사본디스크로 전송된 데이터 스트림이 되며, 해당 작업의 결과는 원본 장치를 비트 단위로 복사한 내용이 담긴 하나의 큰 파일이다. EnCase의 증거 파일인 EWF 포맷 파일은 대상 매체를 비트 단위로 정확하게 보존하며, 비트 단위로 복사된 dd 이미지와 다르게 EWF 포맷 파일은 비트 단위로 복사된 데이터 외에 효율적인 관리를 위해 포장과 꼬리표

Table 1. MD5 and SHA1 Compression functions

Description	MD5	SHA-1
# Round	4	4
# Steps in Round	16 (Total 64 Steps)	20 (Total 80 Steps)
# Input bits	512	512
# Output bits	128	160
Initial Vector	32-bit 4 blocks (A,B,C,D)	32-bit 5 blocks (A,B,C,D,E)

2) 대법원 2007.12.13, 선고, 2007도7257 판결

3) 대법원 2013.07.26. 선고, 2013도2511 판결

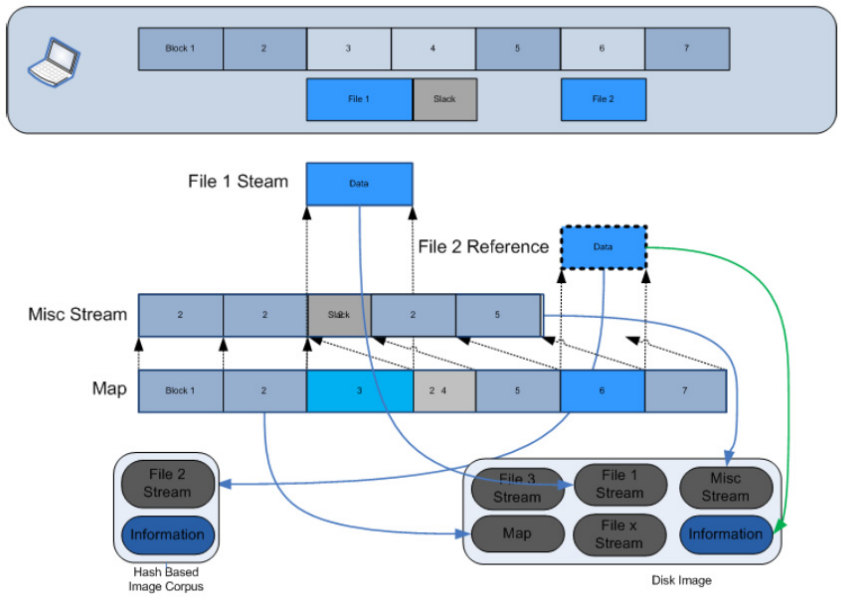


Fig. 1. AFF4 Hash Image Structure

(bag-and-tag) 역할을 하는 메타데이터 같은 정보가 추가된다.

이 정보는 별도의 항목으로 증거파일을 만드는 경우 자동으로 생성되어 이미지에 포함된다. 이렇게 하나의 자동화된 단계를 수행하여 증거가 봉인되기 때문에 당연히 진정성이 성립된 상태로 증거 보관함에 저장되어 증거의 무결성이 보존된다. 그 과정은 그림 Fig. 1과 같으며, 증거 파일을 생성한 후 해당 증거파일이 올바르게 생성되었는지 여부를 판단하기 위하여 재검증 과정을 거친다. 재검증 과정은 이미지 파일을 생성(acquire)하는 과정 이외에도 생성한 이미지 파일을 EnCase에서 실행시키거나 해당 파일의 기록된 해시값을 확인함으로 이미지 파일에 대한 무결성을 체크하여 증거파일이 변조되지 않았음을 확인한다.

III. 관련연구

EWF(Expert Witness Compression Format) 포맷 이외에도 디지털 증거에 대한 이미지 포맷은 지금까지 다양하게 연구되었다. 2006년 DFRWS에서 발표된 "Survey of Disk Image Storage Format"[2]에 의하면 디지털 포렌식의 증거 파일 포맷은 DEB(Digital Evidence Bag), RAW(dd), SMART, ProDiscover 등이 있다. DEB는 '저장장치로부터 추출한 디지털증거를 보관하는 일반적인 컨테이너(Universal Container)' 또는 '모든 형식의 디지털증거를 위한 전자적 포장물(Wrapper)'로 정의하며, 디지털 증거 및 관련 메타데이터를 저장하는 이미지 포맷의 개념이다. DEB의 활용 포맷으로는 EWF, AFF(Advanced Forensic Format)가 존재한다.

AFF는 디스크 이미지와 관련된 메타 데이터를 저장하기 위해 개방적이고 확장 가능한 파일 형식으로 구성된다. AFF 포맷은 EWF와 달리 Container 구조가 공개되어 있으며, 파일 헤더와 Segment로 구성된다. AFF의 파일 헤더는 해당 헤더의 정보, 이름, 메타데이터, Segment 정보로 구성되며 각 Segment는 Page 단위로 관리된다. EWF는 메타데이터가 하나의 섹션에 저장되는 반면, AFF는 각 항목들을 개별 Segment에 저장하는 구조이다[3].

Segment에는 각각의 데이터의 시작 위치와 끝 위치 정보를 담고 있어 원하는 Data 블록을 Scan 할 수 있다. AFF4는 AFF에서 재설계된 버전으로 Map Stream 방식을 사용한다. 이미지 카빙의 결과로 카빙 파일을 복사한 이미지 형식이 출력되며, 이 이미지가 RAID[4]되어 여러 이미지로 나뉜 경우 일반적으로 논리 이미지를 마운트 한 후 사용해야 한다. 만약 하

4) RAID(Redundant Array of Independent Disks): 여러 개의 하드 디스크에 일부 중복된 데이터를 저장하는 기술이다.

Table 2. AFF Segment Structure

Offset	Size	Description
0	4	Segment Header "AFF\00"
4	4	Segment Name Length
8	4	Segment Data Length
12	4	Segment Flag
16		Segment name
…		Segment Data (up to 4GB binary)
…	4	Segment Footer "ATT\00"
…	4	Total Segment Length

나의 복사본이라도 손상된 경우 모든 파일이 손상되어서 사용할 수 없다. AFF의 경우 해당 분리된 이미지가 포렌식 포맷이고 필요한 블록만 사용 가능하다. 특히 AFF4는 Fig.2와 같이 디스크 이미지를 블록 단위로 해시한다. 또한 블록 생략이 가능하며, 생략된 Stream을 모아 해시값을 따로 계산한다[4]. AFF4는 획득 시간과 공간이 절약되고 동일한 이미지에 대해 같은 해시값을 얻는 특징이 있으며 EWF보다 효율적으로 확장된 구조이기 때문에 국외의 경우 EWF와 동등하게 증거물로 사용된다. 하지만 국내에서는 AFF가 프로그램 별로 메타데이터에 대한 명칭이 다르고 호환성에 문제가 있어 증거물로 채택된 판례가 아직까지 없다.

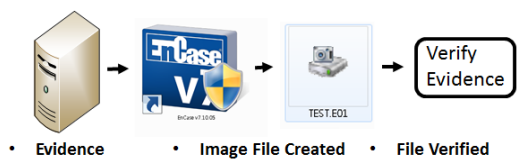


Fig. 2. Disk Imaging Process

EWF 파일 포맷은 고정적인 데이터 구조를 가지고 있어 새로운 항목을 반영하기 어렵고, 폐쇄적인 구조로 어느 프로그램이나 같은 결과를 갖는다. 이러한 특징으로 인해 EWF를 이용한 이미지를 디지털 증거로 채택된 판례가 있어 국내에서는 EWF 포맷이 디지털 증거로 사용된다. 본 논문은 고정적인 구조를 가진 EWF 포맷 파일의 이미징 속도 개선 방안을 제안한다.

IV. EWF 파일 형식

현재 사용하는 포렌식 도구 중 가장 대표적인 도구로 EnCase를 꼽을 수 있다. EnCase는 디스크 이미징 과정 중 해시함수를 선택하고, 해당 해시값을 계산하여 이미지 파일에 대한 무결성을 제공한다. EWF(Expert Witness Format)는 EnCase에서 적용되는 버전 파일의 포맷으로 EnCase v6까지는 E01형식을 사용하며[5], EnCase v7부터는 Ex01 형식을 default로 사용한다[6].

EWF의 파일 구조는 Fig. 3와 같다. 저장매체에서 수집된 정보는 다음과 같이 Chunk라고 불리는 단위와 해당 Chunk의 Checksum(CRC)의 쌍으로 기록된다. Chunk Data는 파일 생성 당시에 크기를 설정 할 수 있으며 기본설정 값은 64섹터이다. CRC의 경우 Adler32[5]를 사용하며 EWF 파일 형식마다 다르게 저장된다. 또한 이미지 파일을 여러 개의 파일로 저장할 경우 파일 구조는 Fig. 4와 같다. 단 Fig. 4는 3개의 이미지 파일을 나타낸다.

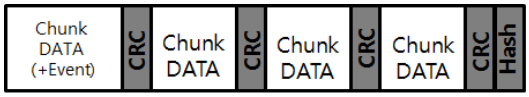


Fig. 3. EWF file Structure

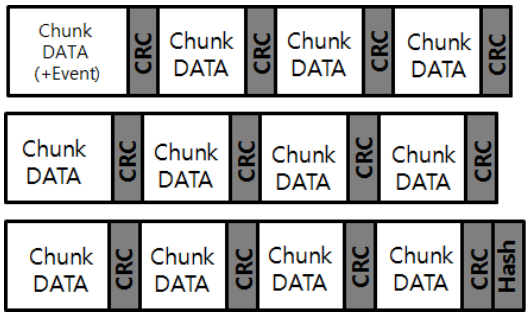


Fig. 4. EWF file Fragment Structure

4.1 E01 파일 형식

파일의 형식은 EnCase v6까지 사용되었다. E01 파일은 Header로부터 시작하며, Table. 3과 같다. 파일의 시작을 나타내는 Signature(45 56

5) Mark adler가 만든 Checksum 알고리즘으로 비트의 변형여부를 확인할 수 있다[7].

Table 3. E01 File Header

Offset	Size	Value	Description
0	8		Signature "EVF\x09\x0d\0a\0ff\0xx00"
8	1	0x01	Start of Fields
9	2		Segment number
11	2	0x0000	End of Fields

46 09 0d 0a ff $00_{(16)}$)와 Fields의 시작과 E01의 파일이 분할될 경우 파일의 개수를 나타내는 Segment Number, 그리고 Fields의 끝으로 이루어져 있다.

Segment number는 사용자가 임의로 선택할 수 있다. Segment number를 m(m≥2)으로 선택하면 이미지 파일은 m개로 나뉘며 각각의 이미지는 개별적으로 사용하는 것이 불가능하고 모든 이미지를 마운트[6]해야 사용이 가능하다. 이미지 파일 생성 시 Image Fragment Size의 값을 설정할 경우 해당 설정한 크기로 이미지를 분할한다.

File Header이후 E01 파일은 섹션 단위로 나뉘어 관리되는데, 섹션 Header 방식은 Table. 4과 같다. E01은 파일을 관리하는 다양한 Header가 존재하며 일반적으로 Header의 종류(Section type Definition), 다음 헤더의 Offset, 해당 Section의 크기(헤더포함)와 0으로 패딩된 값과 Checksum인 Adler32로 구성된다.

Section Type Definition에 정의되는 값들은 E01 파일의 정보가 들어있는 header 섹션, 해당 파일의 볼륨 정보가 들어있는 volume 섹션, 디스크 고유번호, 제품명 등의 원 저장매체에 관한 정보가 있는 Device 섹션, 실제 파일의 정보가 담겨 있는 Sector 섹션과 Table 섹션, 파일이 Segment 된 경우 다음 파일의 존재를 알려주는 Next 섹션, 실제 파일의 끝을 나타내는 Error 섹션, MD5와 SHA-1의 해시 정보가 들어 있는 Digest 섹션, MD5 해시 값을 담고 있는 Hash 섹션, 파일의 끝을 나타내는 done 섹션으로 이루어져 있다. 헤더의 검증 값인 Checksum의 경우 Definition부터 Padding 값까지 Adler32로 헤더를 검증한다.

Table 4. E01 Section Header Definition

Offset	Size	Value	Description
0	16		Section Type Definition
16	8		Next Section Offset
24	8		Section Size
32	40	0x00	Padding
72	4		Checksum (Adler-32)

6) 마운트(mount) : 저장장치에 접근할 수 있는 경로를 디렉터리 구조에 편입시키는 과정, Wikipedia

E01로 이미징한 실제 Data에 대한 검증 값으로 Adler32가 사용된다. 실제 Data는 시작되는 Sector Section 중 volume Section 헤더 이후 데이터를 Adler32를 통해 검증한다. 즉, 64섹터로 기본단위가 설정된 경우 64섹터마다 4바이트의 Adler32(CRC)가 추가되어 이미지가 생성되는 구조이다. 본 CRC는 Table. 3의 Checksum(헤더검증용)과는 다른 입력 값으로 생성된다. 이 값은 파일의 끝까지 64섹터 마다 반복된다.

4.2 Ex01 파일 형식

EnCase v.7 버전부터 사용한 새로운 물리 증거파일 형식인 Ex01은 Fig .5와 같다.

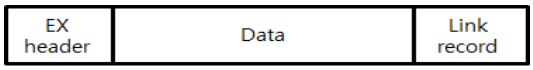


Fig. 5. Ex01 file format

Ex header의 압축, GUID와 서명기능이 추가되고 Data 영역에 섹터 데이터와 엔트리 데이터, 장치 정보가 포함된다. 링크 레코드에는 데이터 영역의 크기와 해시 값, 다음 링크의 기록위치와 파일의 암호화/압축 플래그가 기록된다. Ex01 파일의 시작은 Table. 5와 같다.

파일의 시작을 나타내는 Signature(45 56 46

Table 5. Ex01 File Header (EX header)

Offset	Size	Value	Description
0	8		Signature "EVF2\r\n\x81\x00"
8	1	0x02	Major version
9	1	0x01	Minor version
10	2		Compression method
12	4		Segment file number
16	16		GUID

Table 6. Ex01 Section Header Definition

Offset	Size	Value	Description
0	4		Section Type
4	4		Data Flag
8	8		Previous Section Offset
16	8		Data Size
24	4		Section Descriptor Size
30	4	28	Padding Size
34	28	0x00	Padding
62	4		Checksum (Adler-32)

32 0d 0a 81 00)$_{(16)}$와 2 바이트의 버전 정보와 압축 방식의 정보를 두 바이트로 나타낸다. 0x0일 경우 압축하지 않는 상태이며, 0x1일 경우 LZ 압축 방식을[8], 0x2일 경우 BZip2 방식을 이용한다[9]. 이후 분할된 파일의 개수를 나타내는 Segment file number와 이후 GUID로 이루어져 있다[10]. Ex header 이후에 구성되는 Ex01의 헤더 방식은 Table. 6와 같다.

기존의 E01 파일의 섹션 헤더 방식은 Section Type을 실제 어떤 값이 오는지를 아스키 코드로 정의하였으나 Ex01 방식은 Table. 7과 같이 특정 값이 정의되어 사용된다.

각각의 헤더는 이전 섹션의 Offset 해당 헤더와 데이터의 크기를 나타내고 있다. Section Header가 Previous Section Offset을 가지고 있어 이전 섹션 헤더 위치를 다음 헤더가 가지고 있다. Data Size의 경우 데이터가 0으로 패딩된 값을 포함한다. Section Descriptor Size에 헤더의 사이즈를 나타내고 있으며 Checksum인 Adler32로 헤더를 검증한다.

Table. 7. EX01 Sector Type

value	Description	value	Description
0x01	Device Info	0x09	SHA1 Hash
0x02	Case Data	0x0a	Restart Data
0x03	Sector Data	0x0b	Encryption Key
0x04	Sector Table	0x0c	Memory Extention table
0x05	Error Table	0x0d	Next
0x06	Session Table	0x0e	Final Info
0x07	Increment	0x0f	Done
0x08	MD5 Hash	0x10	Analytical Data

Ex01 파일에도 E01 파일과 마찬가지로 Data에 대한 검증 값으로 Adler32가 사용된다. 검증 절차는 다음과 같다. 먼저 파일을 생성할 때 섹터 단위를 설정한다. Sector Data(0x03) 섹션 이후 설정된 섹터 단위까지의 데이터를 다음에 기록된 4바이트 CRC(Adler32)으로 검증한다. 이 값은 파일의 끝까지 64섹터 마다 반복된다.

V. 제안하는 EWF 이미징 방식

전자 기기가 점점 발전하면서 우리가 사용하는 OS는 '멀티 OS'라고 불릴 만큼 동시에 여러 가지 작업이 가능하다. 이때 각각의 응용프로그램은 하나의 프로세스를 가지며 프로세스 각각은 여러 개의 조각으로 나뉜 쓰레드를 이용한다. 쓰레드는 하나의 프로그램에서 하나의 일을 처리하는 것이 아닌, 동시에 많은 일을 처리할 수 있는 장점이 있고 같은 일을 더 빠른 시간에 처리할 수 있다. 이러한 쓰레드를 이용하여 해시 작업을 처리한다면 이미징 작업에서 해시가 처리하는 시간을 줄일 수 있다.

Fig. 4과 같이 이미지 파일이 두 개 이상인 경우 각각의 이미지 파일은 개별적으로 사용하는 것이 불가능하고 모든 이미지 파일을 마운트한 후에 사용 가능하다. 만약 증거 파일이 있는 이미지가 아닌 다른 이미지 파일이 손상되어 있더라도 증거파일을 사용하지 못하게 된다. 이는 원 파일에 대한 해시값이 변경되어 무결성 검증에 실패하기 때문이다. 이때 증거가 담겨있는 이미지 파일에 대하여 해시값을 가지고 있다면 다른 쪽 이미지 파일이 손상되더라도 증거를 담고 있는 원이미지에 대한 해시값 변경이 없으므로 디지털 증거로 사용이 가능하다.

EWF 파일의 기존 해시 방식은 입력으로 이미지 첫 바이트부터 마지막 바이트까지 순차적으로 사용하기 때문에 병렬 방식이 불가능 하였으나, 이미지를 Segment하여 생성하는 경우에 대하여 각각의 이미

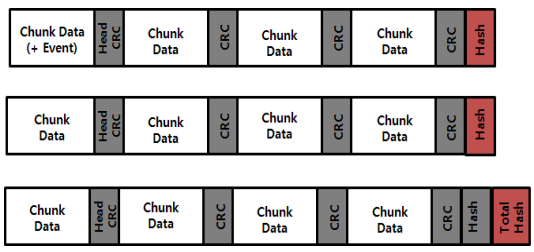


Fig. 6. Our Proposed Hashing Method for EWF files

Table 8. Comparison of our hashing method and others

	Current hashing method for EWF	Our proposed hashing method for EWF	Hashing method for AFF [4]
Implementation method	Serial (all data hashed)	Parallel (all data hashed)	Serial (selected data hashed)
Speed dependency	Depending on total data size	Depending on divided data size	Depending on selected data size
Evidence effect	If any part of image file is damaged, the image file fails for the integrity verification and it cannot be used as a digital evidence.	Even though some parts of image file are damaged, the image file may still be used as a digital evidence.	If any part of the selected data is damaged, the image file fails for the integrity verifications and it cannot be used as a digital evidence.

지 파일에 대해 해시값을 구한다면 충분히 병렬처리 가능하며, OS의 멀티 프로세스를 이용해 여러 개의 해시값을 한 번에 구할 수 있다.

제안하는 각 해시값의 추가 방법은 다음과 같다. Fig. 6와 같이 먼저 각 이미지 파일에 대한 해시값을 계산하여 각 이미지 파일에 덧붙인다. 그 후 각 이미지 파일의 해시값을 입력으로 파일 전체의 최종 해시값을 생성한 후, 마지막 이미지 파일의 끝에 추가한다. 생성된 해시값은 이미지를 재검증하기 위하여 사용한다.

앞서 언급한 것과 같이 제안하는 방법의 장점은 크게 두 가지다. 첫째, 분할된 이미지 파일이 손상된 경우에도 증거 파일을 담고 있는 이미지 파일에 손상이 없다면 분할된 해시값을 통해 무결성을 검증 할 수 있으며, 이는 증거로서 효력을 발휘하게 된다. 둘째, 원 이미지 파일에 대한 해시 속도도 향상된다. 처음부터 마지막까지 순차적으로 입력으로 들어 가야 했던 단일 프로세스를 사용한 기존 방식에 비해 제안 방식은 Segment된 이미지 파일 별로 병렬 처리가 가능하기 때문에 해시값을 생성하는 과정에서 시간적 이득을 볼 수 있다. 또한 마지막 해시값의 입력으로 분할된 이미지 파일의 해시값만 사용하므로 재검증하는 시간에도 무결성을 체크하는 시간을 절약 할 수 있다. 제안하는 방식의 EWF와 기존방식의 EWF, AFF의 해싱방법에 대한 각 장단점은 Table. 8와 같다.

VI. 제안한 해싱 방식에 대한 속도 측정

SIMD(Single Instruction Multiple Data)는 단일의 명령어로 복수의 데이터를 처리하는 개념이다. 한 번의 더하기 연산으로 여러 블록에 대한 각각의 데이터를 처리한다. SIMD 기능은 주로 CPU에서 제공하는 명령어 셋(Instruction set)으로 사용할 수 있다. 그렇기 때문에 CPU마다 지원하는 명령어 셋이 다르며 Intel의 경우 MMX이후 SSE[7] SSE2, SSE3, SSE4, AVX, AVX2 등 많은 버전이 소개되었다. SIMD 기법을 적용하여 앞서 제안한 병렬 구조를 구현하여 시간을 측정하였다. Segment 단위에 따라 4블록, 8블록에 적용하였다. Segment된 파일의 크기는 동일하므로 해시함수의 입력 크기 또한 각 Segment에 대해 동일하다. EWF의 포맷은 SHA-1, MD5를 사용한다는 점 이외에는 공개되어 있지 않다. 해당 실험을 위하여 다음 Table. 9과 같은 환경에서 병렬처리를 이용한 해시함수에 대해 실험하였다.

병렬 처리를 이용한 해시속도를 측정하기 위하여 SHA-1과 MD5의 자체 구현한 소스 방식대로 SIMD중 하나인 SSE2를 4블록, 8블록 단위로 적용하였다. 해시함수의 입력 길이를 기존의 방식에서 1TB를 사용할 때, 제안한 해싱 방식에서는 SIMD가 적용된 4블록 단위의 해시함수의 각 입력은 256GB를, SIMD가 적용된 8블록 단위의 해시함수의 각 입력은 128GB를 사용하였다. 그 결과는 Table. 10과 같다.

SHA-1은 SIMD를 사용할 경우 기존의 방식보다 4블록의 경우 2.12배의 시간 단축이 되고, 8블록의

Table 9. Test Environment

CPU	OS	Ram
Intel Core i7-4790u 3.60GHz	Win 7 64bit	16G

7) Streaming SIMD Extensions

Table 10. Parallel Processing Imaging Hash Times

	Hash Function	Mb/Sec	1TB Time	Rate (C/P)
Current Method(C)	SHA-1 (Serial)	253.2	69.0m	-
Proposed Method(P)	SHA-1 (4-Block)	559.6	31.2m	2.2
Proposed Method(P)	SHA-1 (8-Block)	1317.1	13.3m	5.2
Current Method(C)	MD5 (Serial)	416.4	42.0m	-
Proposed Method(P)	MD5 (4-Block)	1374.5	12.7m	3.3
Proposed Method(P)	MD5 (8-Block)	4876.1	3.6m	11.7

경우 5배의 시간 단축이 되었다. 마찬가지로 MD5의 경우 SIMD를 사용할 경우 기존의 방식보다 4블록 단위에서는 3.3배의 시간 단축이 되며, 8블록 단위에서는 11.7배의 시간 단축이 되었다. 해당 결과는 입출력이나 다른 영향을 줄 수 있는 요소를 배제하고 오로지 암호학적 관점으로 해시함수만 이용한 결과로 실제 상황에 적용할 경우 환경에 따라 다른 결과를 얻을 수 있다. 2016년 현재 주로 사용하는 인터페이스인 SATA[8] 혹은 USB[9] 포트의 경우 I/O 속도가 해시함수보다 느려 입출력에 영향을 받는 이미징 과정에 해시함수의 속도가 미치는 영향이 적지만 차세대 인터페이스 방식인 Thunderbolt[10]나 PCI Express[11]방식의 경우 충분히 I/O속도가 빨라지므로 제안하는 방식이 무결성 검증 과정에서 병렬로 처리한 만큼의 속도 향상이 가능하다.

VII. 결 론

본 논문에서는 암호학적 관점에서의 이미징 효율성 개선 방안을 연구하였다. 무결성 검증을 위한 소요시간을 개선시키기 위하여 EnCase에서 사용하는 EWF 파일에 대해 병렬 처리를 이용한 해싱 방법을 제안하였다.

8) Serial ATA : 하드 디스크를 위한 고속 인터페이스, SATA 3의 최대속도는 750MB/s
9) Universal Serial Bus : 범용 직렬 버스 USB 3.0의 최대 속도는 625MB/s
10) Thunderbolt : MAC 기반의 차세대 인터페이스, Thunderblot 2의 최대속도는 2.5GB/s
11) 메인보드에 그래픽카드, 사운드카드 같은 카드를 사용하기 위해 만들어진 규격, PCI Express 3.0의 경우 최대속도는 31.5GB/s

EWF에서 지원하는 Segment 기능을 이용하여 증거 이미지를 분리하는 경우, 분리된 이미지 중 하나라도 손상이 있다면 기존 방식에서는 전체 이미지에 대한 해시값만 저장하기 때문에 증거가 있는 부분의 이미지 파일에 손상이 없어도 증거파일로 사용할 수 없었다. 하지만, 본 연구에서 제안한 방법은 Segment된 이미지마다 해시값을 병렬 방식으로 추가적으로 구하므로 각 이미지 파일에 대한 손상 여부를 알 수 있으며 손상되지 않는 부분에 대하여 무결성 검증을 할 수 있다. 또한, 제안한 방식은 병렬처리가 가능해 무결성 검증 시간을 단축시킬 수 있다.

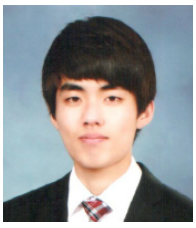
References

[1] P. Deutsch, "GZIP file format specification version 4.3," RFC 1952, May. 1996.
[2] "Survey of Disk Image Storage Formats," DFRWS 2006, Sep, 2006,
[3] S. Garfinkel, "Advanced Forensic Format : An Open, Extensible Format For Disk Imaging", Advances in Digital Forensic, FIP, "http://nrs.harvard.edu/urn3:HUL.InstRepost:2829932"", Jan, 2006
[4] B, Schatz, "Wirespeed : Extending the AFF4 forensic container format for scalable acquisition and live Analysis ," DFRWS 2015 USA, vol. 14, pp. 45-54, Aug. 2015
[5] Joachim Metz, "Expert Witness Compression Format specification," GFDL, Mar, 2013
[6] Joachim Metz, " Expert Wintess Compression Format version 2 specification," GFDL, Aug, 2012
[7] E.-h. Yang and J. C. Kieffer. "On the redundancy of the fixed database Lempel-Ziv algorithm for mixing sources," IEEE Trans. Inform, Theory, vol.43, pp.1101-1111, Jul, 1997
[8] Stone, et, al "Stream Control Transmission Protocol (SCTP) Checksum Change", RFC3309, Sep, 2002
[9] Julian Seward, "bzip2 and libbzip2, ver-

sion 1.0.5, Aprogram and library for data compression", GFDL, Sep, 2008
[10] GUID structure (windows)", https://msdn.microsoft .com/ko-kr/library/windows/desktop/aa373931(v=vs.85).aspx, Aug, 2012

〈저자소개〉

신 용 학 (Yonghak Shin) 학생회원
2015년 2월: 국민대학교 수학과 졸업
2015년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식



김 도 원 (Dowon Kim) 학생회원
2015년 2월: 국민대학교 수학과 졸업
2015년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식

이 창 훈 (Changhoon Lee) 종신회원
2001년 2월: 한양대학교 수학과(이학사)
2003년 2월: 고려대학교 정보보호대학원(공학석사)
2008년 2월: 고려대학교 정보보호대학원(공학박사)
2009년 3월~2011년 2월: 한신대학교 컴퓨터공학부 전임강사
2011년 3월~2012년2월: 한신대학교 컴퓨터공학부 조교수
2012년 3월~현재: 서울과학기술대학교 컴퓨터공학과 조교수
〈관심분야〉 정보보호, 암호학, 디지털포렌식, 컴퓨터이론

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 학사/이학석사
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC 정보보호 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~현재: 국민대학교 수학과 부교수
2014년 3월~현재: 국민대학교 일반대학원 금융정보보안학과 부교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식