Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.6, Dec. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.6.1373

윈도우 서버 2012에서 데이터 중복 제거 기능이 적용된 파일의 복원 방법에 관한 연구

손 관 철,† 한 재 혁, 이 상 진‡
고려대학교 정보보호연구원

A Study of Method to Restore Deduplicated Files in Windows Server 2012

Gwancheol Son,† Jaehyeok Han, Sangjin Lee‡
Institute of Cyber Security & Privacy (ICSP), Korea University

요 약

중복 제거는 데이터를 효과적으로 관리하여 저장 공간의 효율성을 높이기 위한 기능이다. 중복 제거 기능이 시스템에 적용되면 저장되어 있는 파일을 청크 단위로 분할하고 중복되는 부분은 하나의 청크로만 저장함으로써 저장 공간을 효율적으로 사용할 수 있게 한다. 하지만 중복 제거된 데이터에 대해 상용 디지털 포렌식 도구에서 파일시스템 해석을 지원하지 않으며, 도구로 추출된 원본 파일을 실행하거나 열람할 수 없는 상황이다. 따라서 본 논문에서는 중복 제거 기능을 적용할 수 있는 윈도우 서버 2012 시스템을 대상으로 청크 단위의 데이터를 생성하는 과정과 그 결과로 생성되는 파일의 구조를 분석하고, 기존 연구에서 다뤄지지 않은 청크가 압축되는 경우에 대해서도 분석결과를 도출하였다. 이러한 결과를 바탕으로 디지털 포렌식 조사에서 적용할 수 있는 수집 절차와 원본 파일로 재조합하기 위한 방법을 제시한다.

ABSTRACT

Deduplication is a function to effectively manage data and improve the efficiency of storage space. When the deduplication is applied to the system, it makes it possible to efficiently use the storage space by dividing the stored file into chunks and storing only unique chunk. However, the commercial digital forensic tool do not support the file system analysis, and the original file extracted by the tool can not be executed or opened. Therefore, in this paper, we analyze the process of generating chunks of data for a Windows Server 2012 system that can apply deduplication, and the structure of the resulting file(Chunk Storage). We also analyzed the case where chunks that are not covered in the previous study are compressed. Based on these results, we propose the method to collect deduplicated data and reconstruct the original file for digital forensic investigation.

Keywords: Deduplication, Deduplicated File Restoration, Chunk Storage, $REPARSE_POINT, System Volume Information

I. 서 론

IT 기술의 발전으로 디지털 장치가 저장할 수 있는 공간이 증가하고 있지만, 생성되는 데이터의 총량은 더 빠르게 증가하고 있으므로 미래의 시스템은 저장된 데이터를 더 효과적으로 관리해야 한다[1]. 이러한 요구사항에 대응하여 마이크로소프트는 윈도우 서버 2012부터 저장 공간을 효율적으로 관리하기 위한 데이터 중복 제거 기능을 지원하고 있으며, 이는 NTFS(New Technology File System)에서 확

Received(09. 26. 2017), Modified(11. 15. 2017), Accepted(11. 16. 2017)
† 주저자, songc13@gmail.com
‡ 교신저자, sangjin@korea.ac.kr(Corresponding author)

장된 기능으로 동작한다[2]. 저장 장치 제작 기술의 발전이 물리적인 측면에서 저장 공간을 확보하기 위한 방향이었다면, 데이터 중복 제거 기술은 논리적인 측면에서 저장 공간을 확보한다는 점에서 미래지향적이며 앞으로의 전망이 더욱 기대되는 기술이다.

시스템에 중복 제거 기능이 적용되면, 기존의 저장 방식과 다르게 원본 파일을 청크 단위로 분할하여 논리적인 형태의 청크 저장소에 기록한다. 청크 저장소는 시스템이 개별적으로 관리하기 때문에 일반 사용자가 식별할 수 없다. 게다가 현실은 중복 제거된 데이터에서 원본 데이터를 확보할 수 없기 때문에 기존의 수집 방법으로 추출된 파일을 실행하거나 열람할 수 없다. 이러한 이유로 디지털 포렌식 조사 시 수집 과정에서 중복 제거 데이터 수집에 대한 제한사항이 존재한다. 중복 제거 기능이 설정되어 있으면 EnCase, FTK, X-Ways Forensics와 같이 주로 사용하는 상용 디지털 포렌식 도구도 디렉터리 구조만 확인할 수 있고, 원본 데이터를 논리적으로 추출하여 복사하는 기능은 지원되지 않고 있다.

본 논문에서는 중복 제거 시스템을 이해하고, 데이터 중복 제거 기능이 적용된 마이크로소프트의 윈도우 서버 2012 시스템을 분석하는데, 중복 제거 기능에 의해서 생성된 청크 저장소를 분석하고 시스템이 데이터를 관리하는 원리를 파악한다. 연구 결과를 바탕으로 데이터 중복 제거 기능이 활성화된 볼륨의 식별 방법과 식별된 볼륨에서 중복 제거 데이터를 수집하는 방법을 제시한다. 그리고 수집한 데이터로부터 중복 제거 파일을 복원하는 방법을 제시한다.

II. 관련 연구

최초의 중복 제거 시스템은 네트워크상에서 통신망의 부하를 줄이기 위한 알고리즘 개발로부터 시작되었으며[3], 이후 알고리즘의 효율성을 평가하는 연구[4]와 관련된 기술이 개발되면서 클라우드 저장소에 적용되기 시작하였다. 클라우드 환경에서 사용자 간의 데이터가 누출될 수 있는 위험을 줄이면서 중복 제거 기능을 활용하는 메커니즘도 연구되었다[5]. 또한 중복 제거가 활성화된 시스템에서 중복 제거 기능이 적용된 데이터를 식별하기 위한 필요성을 강조하고 있다[6]. 특히 Dario Lanterna 등[2]은 중복 제거 시스템의 내부 데이터 저장구조를 개괄적으로 분석하였으며, 디지털 포렌식 관점에서 데이터 수집을 위한 방법의 필요성과 원본 데이터를 복원하기 위한 기술 개발의 필요성을 언급하고 있다.

스마트폰 플래시 메모리의 페이지에 기록된 데이터가 수정될 경우에는 새로운 페이지가 생성되는데 이 과정에서 중복된 페이지가 증가한다. 이러한 중복 페이지를 효율적으로 분석하기 위한 방법으로 중복 제거가 활용되었다[7]. 해시값 비교를 통해서 중복된 파일을 선별함으로써 수집한 데이터로부터 조사관이 분석해야 할 대상의 범위를 축소시키는 연구도 있다[8,9]. 이렇게 최근에는 기술 개발보다 디지털 포렌식 관점에서 조사의 효율성과 한계점을 보완하기 위한 방향으로 중복 제거 기술과 관련된 연구가 진행되고 있다.

하지만 중복 제거된 데이터를 중복되기 이전 데이터(원본 파일)로 복원하여 운영체제 또는 응용 프로그램에서 읽을 수 있도록 하여 조사관이 열람할 수 있는 형태로 가공하는 방법에 대한 연구는 진행된 사례가 없다. Dario Lanterna 등[2]은 중복 제거된 데이터로 구성된 청크 저장소를 대략적으로 분석했기 때문에 실제 운용환경에서 추출한 데이터의 구조와 비교했을 때 일치하지 않는 내용이 다수 확인되었으며, 이는 충분한 실험 데이터를 대상으로 구조 분석이 진행되지 않았던 것으로 보인다. 따라서 디지털 포렌식 조사 과정에서 활용할 수 있는 경우도 제한적일 수밖에 없다. 구체적으로 해당 연구의 미흡한 점은 다음과 같다. 먼저, 중복 제거된 파일을 식별하는 방법이 명확하지 않고 특정 스트림 파일을 식별하는 방법에 대한 분석 결과에 오류가 존재한다. 특정 데이터 파일을 찾아가는 방법의 효율성이 떨어지며 청크 해시 알고리즘에 대한 분석이 부정확하다. 또한, 청크가 압축된 경우 압축 알고리즘을 파악하지 못했으며 재조합하는 과정에 적용하지 못하였다. 따라서 본 논문에서는 기존 연구에서 확인한 미흡한 연구내용을 보완하고, 추가적으로 분석된 구조를 살펴본다. 그리고 이러한 연구결과를 바탕으로 원본 파일로 재조합하여 조사관이 파일 내용을 조사할 수 있게 함으로써 디지털 포렌식 조사에 활용할 수 있도록 한다.

III. 데이터 중복 제거 기능과 본 연구의 의의

본 장은 윈도우 서버 2012의 '데이터 중복 제거'에 대하여 간략하게 소개하고 디지털 포렌식 조사 과정에서 주로 사용하는 상용 디지털 포렌식 도구의 한계점에 대하여 3.1절에서 구체적으로 설명한다. 또한 '데이터 중복 제거'로 생성된 데이터를 상세히 분

석하고, 분석 결과를 바탕으로 Dario Lanterna 등[2]의 연구 결과와 비교하여 개선된 내용을 3.2절에 기술한다.

3.1 배경지식 및 상용 디지털 포렌식 도구의 한계

일반적으로 중복 제거 시스템은 파일을 청크라는 작은 단위로 분할한다. 분할된 청크에서 개별적으로 해시값을 생성하며, 생성된 해시값과 해시 저장소에 저장된 해시값을 비교하여 일치하지 않다면 데이터 저장소에 새로운 청크가 저장되고, 모든 청크의 해시값은 해시 저장소에 추가된다.

중복 제거 시스템은 처리방식에 따라서 인라인(inline) 처리 방식과 포스트(post) 처리 방식으로 나뉜다. 그리고 청크의 크기를 결정하는 알고리즘은 고정길이 알고리즘과, 가변 길이 알고리즘이 있다[2].

윈도우 서버 2012에서 중복 제거 기능을 사용하려면 서버 관리자에서 '데이터 중복 제거' 기능을 설치하고, 사용자가 원하는 볼륨을 추가적으로 선택하여 설정해야 한다. '데이터 중복 제거'는 포스트 처리 방식으로 동작하고, Rabin's Fingerprints 알고리즘[10]을 기반으로 청크의 크기를 결정하는 가변 길이 알고리즘이 적용되었다. 또한 생성되는 청크의 길이를 32KB~128KB로 제한하여 가변 길이 알고리즘의 단점을 보완했다[11].

'데이터 중복 제거'로 생성되는 청크 저장소는 공통적으로 'System Volume Information' 폴더 하위에 구성되며 'Dedup' 폴더가 추가로 생성되어 Fig.2.와 같은 디렉터리 구조를 갖는다. 'ChunkStore' 폴더 하위에 생성된 '{A725CCCF-DEE2-49F3-8F00-F4549650B69B}.ddp' 폴더의 이름은 청크 저장소의 식별자와 같다. 'Stream' 폴더에는 청크 데이터의 해시값이 파일 형태(.ccc)

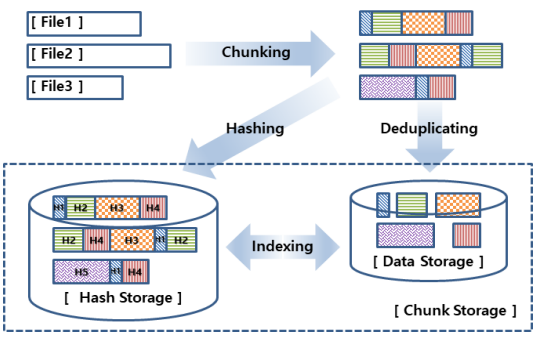

Fig. 1. Operation Concept of Deduplication

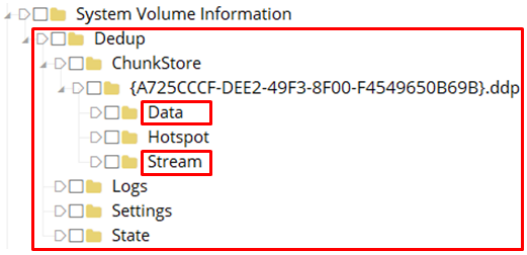

Fig. 2. Directory Organization of ChunkStore

로 저장되고 'Data' 폴더에는 청크 데이터가 파일 형태(.ccc)로 저장된다.

서버는 실시간으로 서비스를 지원해야 하는 특성으로 인하여, 주로 활성 상태에서 데이터를 수집하게 된다. 그러나 EnCase, FTK, X-Ways Forensics와 같이 주로 사용하는 상용 디지털 포렌식 도구는 활성 상태의 수집 대상으로부터 중복 제거 파일을 논리적으로 추출하여 복사하는 기능을 지원하지 않고 있다. NTFS는 MFT Entry의 $DATA에서 중복 제거 파일을 Sparse 속성으로 관리하기 때문에 일반 파일을 논리적으로 추출하는 디지털 포렌식 도구의 기능으로 중복 제거 파일을 복사하여 열람하면, 모든 데이터가 '0x00'로 채워져 있다. Fig.3.은 EnCase v8.05로 해당 파일을 열람했을 때의 모습이고, FTK와 X-Ways Forensics에서도 동일하다.

선별 수집을 위한 키워드 검색에서도 정상적인 결과를 얻을 수 없다. 바이너리 데이터는 데이터(data) 파일(.ccc)에서 검색될 수 있지만 정확히 어떤 원본 파일의 바이너리 데이터인지 확정할 수 없다. 문서나 압축 파일과 같이 원본 콘텐츠를 인코딩 또는 압축하여 저장하는 파일의 내용은 키워드 검색

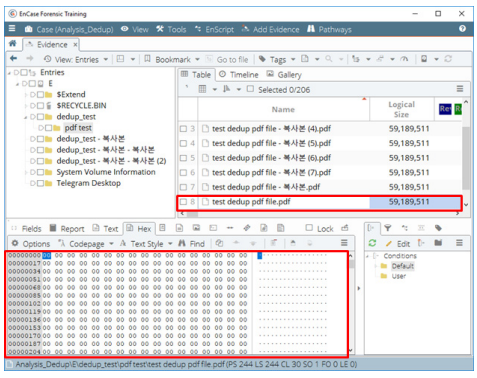

Fig. 3. Hex Values of Deduplicated Files (EnCase v8.05)

으로 발견할 수 없기 때문에 문제가 된다. 따라서 중복 제거 파일을 조사하려면, 분할된 청크를 찾고 재조합해야 한다.

3.2 데이터 중복 제거 분석 및 발전 사항

본 절에서는 중복 제거 파일을 복원하기 위하여 MFT Entry와 스트림(stream) 파일(.ccc), 데이터 파일(.ccc)의 분석 결과를 설명한다.

MFT Entry는 중복 제거 파일을 식별하고, 특정 스트림 파일의 위치 정보를 얻는 것을 목적으로 참조한다. 파일이 중복 제거되면, 해당 파일의 MFT Entry는 $REPARSE_POINT 속성을 포함한다. $REPARSE_POINT 속성은 사용자가 파일 열기를 요청할 때 참조된다. 파일시스템은 파일 열기 동작을 수행하기 위해서 $REPARSE_POINT 속성 내의 Reparse Type Flag 값을 식별한다. 그러고 나서 식별된 값에 따라 해당 데이터 포맷을 처리할 수 있는 파일시스템 필터로 리다이렉트 한다[12].

윈도우 서버 2012는 해시값을 원본 파일의 순서에 따라서 해시 스트림으로 관리하며, 해당 해시 스트림을 'Stream' 폴더에 파일 형태로 저장한다. 해시 스트림은 원본 데이터를 복원하는 과정에서 각 청크의 순서를 맞추기 위해서 필요하므로 반드시 획득되어야 한다. 또한 시스템이 해시값과 청크를 따로 관리하기 때문에 분할된 청크 데이터가 저장된 데이터 파일의 위치 정보를 얻는다.

마지막으로 데이터 파일에서는 원본 파일로 재조합하기 위해서 필요한 청크 데이터를 추출한다. 청크 데이터는 'Data' 폴더에 파일 형태로 저장된다. 하나의 파일에 다수의 청크를 저장할 수 있으며, 각 파일의 크기는 약 100MB를 넘지 않는다.

Fig.4.는 앞서 설명한 내용을 바탕으로 청크 데이터를 획득하여 원본 파일을 복원하는 과정을 표현한 것으로 Fig.4.에서 제시한 흐름에 따라 직접 중복 제거된 예제 파일을 분석하면서 개선시킨 분할된 청크 획득 과정을 3.2.1절에서 설명하고, 3.2.2절에서는 제시한 청크 획득 과정에서 Dario Lanterna 등 [2]의 연구보다 개선된 내용을 상세히 서술한다.

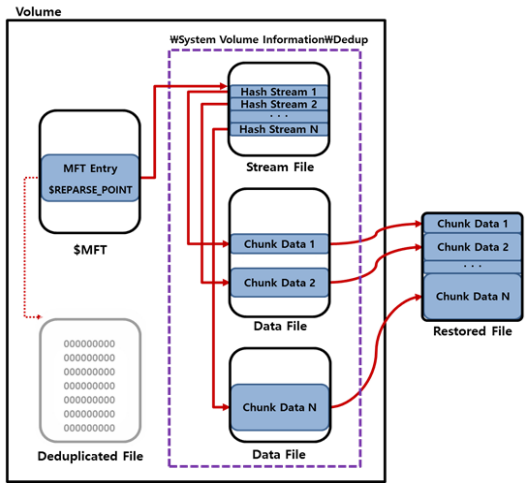

Fig. 4. Process to acquire the chunks for restoration

3.2.1 분할된 청크 획득 과정

분할된 청크를 획득하려면 가장 먼저 중복 제거된 파일이 식별되어야 한다. $REPARSE_POINT 속성(0xC0)에서 해당 속성의 크기가 0xA0이고 Reparse Type Flag 값이 0x80000013일 때 중복 제거된 파일로 식별할 수 있다.

Fig.5.는 분석 예제인 'test dedup pdf file.pdf'라는 중복 제거 파일의 MFT Entry를 분석한 것이다. 분석된 모든 필드에 대한 정보는 Fig.4.에서 얻을 수 있다. $REPARSE_POINT 속성 내에서 0x38 오프셋부터 16바이트 값은 청크 저장소의 식별자 정보이다. 청크 저장소는 여러 개가 존재할 수 있는데, 청크 저장소의 식별자는 중복 제거된 파일의 해시 스트림과 청크 데이터가 어떤 청크 저장소에 저장되었는지 구별하는 목적으로 참조된다. Fig.2.에서 Fig.5.에 표시한 청크 저장소의 식별자 정보와 동일한 값을 가지는 '{A725CCCF-DEE2-49F3-8F00-F4549650B69B}.ddp' 폴더를 찾을 수 있다.

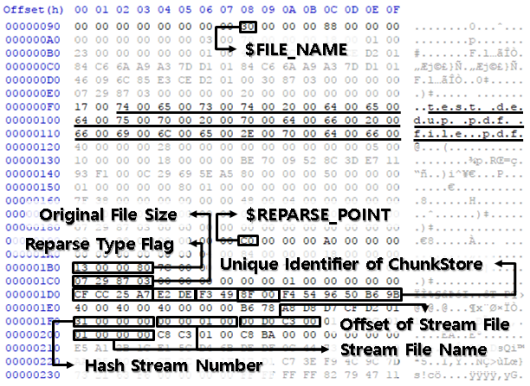

Fig. 5. MFT Entry Analysis of a Deduplicated File

MFT Entry를 분석하면 중복 제거 파일을 식별할 수 있을 뿐만 아니라 해시 스트림에 대한 위치 정보도 함께 획득할 수 있다. Fig.5.의 $REPARSE_POINT 속성 내에서 0x5C와 0x68 오프셋의 4바이트 값을 서로 붙이면 'Stream' 폴더에 저장된 파일 이름(00010000.00000001)이다. 또한 0x58 오프셋부터 4바이트 값은 해시 스트림 번호(0x31)이고, 0x60 오프셋부터 4바이트 값은 스트림 파일의 오프셋(0xC3D000)이다. 분석한 세 가지 정보를 종합하면, Fig.6.과 같이 'Stream' 폴더에 저장된 '00010000.00000001.ccc' 파일의 0xC3D000 오프셋에서 동일한 해시 스트림 번호(0x31)를 갖는 원본 파일의 해시 스트림 정보를 획득할 수 있다.

Fig.6.은 해시 스트림 정보를 저장하고 있는 'Ckhr' 섹션의 구조를 분석한 것이다. 'Ckhr' 섹션에는 모두 동일한 오프셋(0x68)에서 8바이트 크기의 시그니처(53 6D 61 70 01 04 04 01)를 시작으로 Smap 영역이 시작된다. Smap 영역은 청크의 해시값과 청크가 저장된 위치 정보를 한 쌍으로 구성한 구조체 배열이다. 각 구조체에 할당된 크기 S_{Stream}은 0x40이다. 'Smap' 영역의 크기 값 S_{Smap}에서 시그니처 크기(0x8)만큼을 제외하고, S_{Stream} 값(0x40)으로 나눠서 해시 스트림 구조체의 개수를 구할 수 있다. 계산된 구조체의 개수는 중복 제거 파일 복원을 위해서 재조합해야 할 청크 개수와 같다. 따라서 재조합해야할 청크 개수 N_{Chunk}를 구하는 식은 (1)과 같다.

$$N_{Chunk} = \frac{S_{Smap} - 8}{S_{Stream}} \quad (1)$$

Fig.7.은 분석된 스트림 구조체의 각 필드 값을 정리한 것이다. 청크 번호는 새로운 청크가 저장소에 추가될 때마다 1씩 증가한다. 누적된 청크 크기는 첫 번째 청크 크기부터 현재까지의 청크의 크기를 합산한 값이다. 앞서 언급한 값들은 청크를 재조합하는 과정이 정상적으로 진행되고 있는지 점검할 수 있는 검증용 데이터로 활용될 수 있다. 스트림 구조체에서 분할된 청크를 찾기 위한 핵심 정보는 데이터 저장소의 파일 이름, 데이터 저장소 파일에서 오프셋, 청크의 해시값이다. Fig.6.의 Stream 1을 예로 들면, 'test dedup pdf file.pdf' 파일의 첫 번째 청크는 'Data' 폴더 내의 '00000007.00000001.ccc' 파일에 저장되어 있다. 해당 파일의 오프셋 0xC82600에는 Fig.8.과 같이 청크 데이터를 관리하는 또 다른 'Ckhr' 섹션이 존재한다.

Fig.8.은 'Ckhr'이라는 시그니처를 시작으로 0x8 오프셋부터 4바이트 값에 앞서 참조한 해시 스트림 구조체의 청크 번호와 동일한 값이 있고, 다음 4바이트 크기의 0xC 오프셋에서 해당 섹션이 저장하고 있는 청크 데이터의 크기 값이 있다. 0x28 오프셋 위치부터 0x20 바이트 크기만큼의 데이터는 해당 섹션이 저장하고 있는 청크의 해시값이다. 청크 번호와 청크 크기, 청크 해시값은 스트림 구조체 Stream 1의 청크 번호, 청크 크기, 청크 해시값과 동일하다. 즉 3가지 값이 서로 같은지 비교하면 파싱하려는 청크 데이터가 올바른지 검증할 수 있다.

분석에 사용했던 예제 파일은 확장명에서 유추할 수 있듯이 PDF 파일 포맷으로 저장했다. 'Ckhr' 섹션 내의 0x58 오프셋은 해시 스트림 구조체 Stream 1이 가리키던 원본 데이터의 첫 번째 청크

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Chunk Number				Data File Name1				Offset of Data File				Data File Name2			
Cumulative Chunk Size				Unknown											
Hash Value of Chunk (0x20bytes)															
								Chunk Size				Unknown			

Fig. 7. Hash Stream Structure of Smap

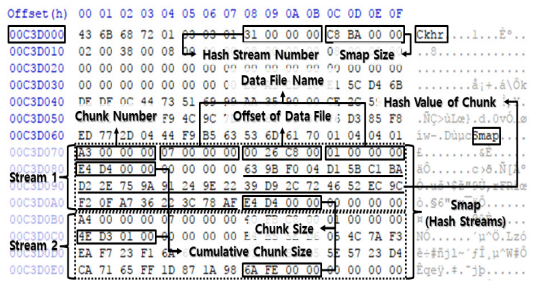

Fig. 6. Stream File(.ccc) Analysis

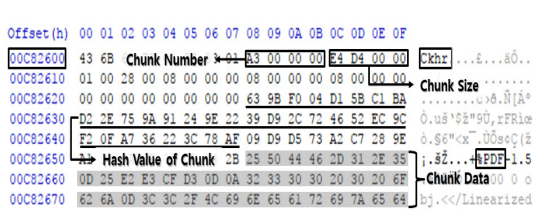

Fig. 8. Data File(.ccc) Analysis

데이터가 시작되는 위치로써 '%PDF'라는 PDF 파일 포맷의 시그니처가 존재한다. 다른 스트림 구조체가 참조하는 'Ckhr' 섹션도 오프셋 0x58 위치에서 청크 데이터가 시작된다. 따라서 중복 제거된 'test dedup pdf file.pdf' 파일의 원본 데이터를 복원하려면, 식 (1)의 결과로 도출된 0x2EB개의 청크가 모두 재조합되어야 한다.

3.2.2 분할된 청크 획득 과정의 개선된 사항

3.2.2.1 청크 획득을 위한 주요 파일 식별 방법 개선

본 논문에서는 MFT Entry의 $REPARSE_POINT 속성을 분석하여 NTFS가 중복 제거 파일을 관리하는 원리를 파악하고, 데이터 중복 제거 기능이 적용된 파일을 식별하는 방법을 제시 했다.

'Stream' 폴더에서 복원하려는 중복 제거 파일의 해시 스트림이 저장되어있는 특정 '.ccc' 파일을 식별하기 위해서는 $MFT의 MFT Entry 분석을 통해 $REPARSE_POINT(0xC0) 속성 정보를 확인해야 한다. $REPARSE_POINT 속성 내에서 0x78 오프셋부터 고정 크기(30바이트)의 값을 Dario Lanterna 등[2]의 방법으로 스트림 파일 내에 저장된 원본 파일의 해시 스트림 정보로 식별할 경우에 정상적으로 중복 제거 파일을 복원할 수 없는 경우가 발생하였으며, 실험한 결과로는 평균적으로 약 7.5%의 데이터만 복원이 가능하였다. 이러한 결과는 $REPARSE_POINT 속성 정보의 구조 분석이 명확히 되지 않고 특정 샘플 데이터에 대해서만 분석을 진행하였기 때문이다. 다양한 샘플 데이터로 분석한 결과를 바탕으로 해시 스트림 정보를 식별하는 용도로 사용하던 필드의 크기는 가변적임을 확인하였다. 따라서 본 논문에서 제시한 청크 획득 방법은 해당 영역의 데이터를 참조하지 않고 원본 데이터를 복원한다.

Dario Lanterna 등[2]의 방법으로 해시 스트림 구조체에서 얻은 청크 해시값을 가지고 청크 데이터가 저장된 특정 '.ccc' 파일과 'Ckhr' 섹션을 찾기 위하여 모든 데이터 파일을 대상으로 바이너리 검색을 하는 것은 효율성이 떨어진다. 본 논문은 이를 개선하기 위하여 3.2.1절에서 해시 스트림 구조체를 완벽하게 분석하였으며, 해당 구조체 내에서 데이터 파일명과 파일 내의 오프셋 정보를 참조하여 찾으려는 청크가 저장된 'Chkr' 섹션에 접근한다.

3.2.2.2 청크 압축 및 해시 알고리즘에 대한 분석 결과

윈도우 서버 2012에서는 볼륨의 저장 가능 공간을 더 확보하기 위하여 추가적으로 청크를 압축하는 경우가 존재한다. 그러나 아직까지 압축 대상 청크로 선택되는 조건과 청크 데이터를 압축하기 위해서 사용하는 압축 및 압축 해제 알고리즘은 공개되지 않았다. 또한 청크 데이터를 압축 및 압축 해제하는 알고리즘에 대해서는 'LZNT1'가 기술적으로 더 발전된 형태인 'LZNT1+'일 것이라고 추측만 했고, 데이터 파일의 'Ckhr' 섹션 내에서 오프셋 0x5C 위치부터 청크 데이터가 저장되는 것으로 연구되었다[2]. 해당 연구 결과는 청크 데이터가 압축되는 경우를 고려하지 않고 분석한 결과이다. 청크 데이터가 압축되면, 압축 해제를 고려하여 청크 시작위치에 4바이트 크기의 압축 해제를 위한 정보가 추가된다. 따라서 데이터 파일의 'Ckhr' 섹션 내에서 오프셋 0x58부터 데이터를 파싱하고, 청크 데이터를 압축 해제하여 원본 데이터로 복원해야 한다.

Fig.9.는 청크 데이터가 압축되었을 때 데이터 파일의 'Ckhr' 섹션을 분석한 것이다. 0x20 오프셋에 압축 여부를 확인할 수 있는 플래그 정보가 존재한다. 해당 값이 0x2일 때, 0x58 오프셋부터 저장된 청크 데이터는 압축된 것을 의미한다.

압축된 청크 데이터에서 원본 데이터를 얻기 위하여 사용자의 요청으로 중복 제거 파일의 원본 데이터를 불러오는 과정을 분석했고, 청크 데이터에 적용된 압축 및 압축 해제 과정을 파악하였다. 분석 결과 윈도우 서버 2012는 청크 데이터를 압축할 때 Plain LZ77 압축 및 압축 해제 알고리즘[13]을 사용한다. 추출한 압축된 청크 데이터에 Plain LZ77 압축 해제 알고리즘을 적용하면 Fig.10.과 같이 원본 청크 데이터를 얻을 수 있다. 예제로 사용한 파일은 압축된 데이터와 압축 해제된 데이터의 명확한 차이를 비교하기 위하여 임의로 선정한 실행 파일 앞부분에 'abc'문자열이 반복되는 300바이트 크기의 바이너리 데이터를 삽입한 것이다.

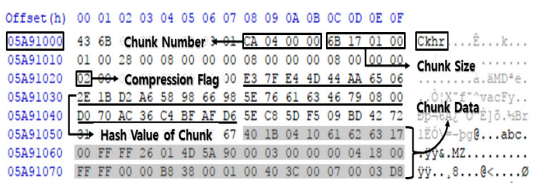

Fig. 9. Compression Flag of the Chunk Data

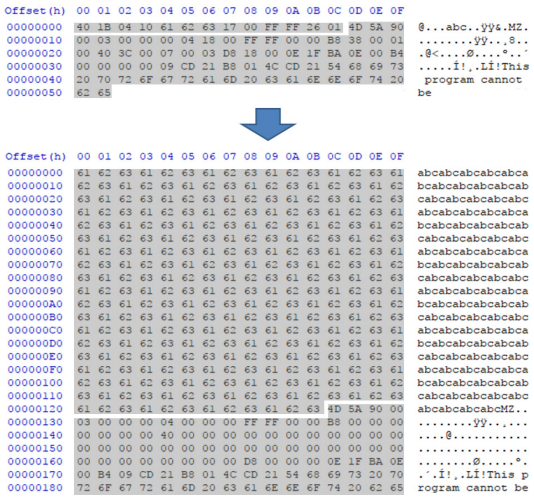

Fig. 10. Plain LZ77 Decompression Example of the Compressed Chunk data

청크 압축 알고리즘뿐만 아니라 청크 해시값 필드에 적용되는 해시 알고리즘도 'SHA-256'라고 잘못 추측하고 있다[2]. 공개된 몇 가지 해시 알고리즘에 대해서 테스트만 해봐도 적용된 해시 알고리즘은 'SHA-512'이고, 청크 저장소를 구성하기 위하여 해당 해시 알고리즘으로 생성된 값에서 상위의 32바이트만 사용한다.

IV. 중복 제거 데이터 수집 및 복원 알고리즘

4.1 중복 제거 데이터 수집 방법

현재의 기술로 중복 제거 기능이 적용된 시스템에서 데이터를 수집하는 방법은 두 가지 방법이 있다. 첫째는 수집 대상이 비활성 상태일 경우에 시스템 전체를 복제하는 Disk to Disk Copy 방법이다. 둘째는 수집 대상이 활성 상태일 경우에 데이터를 선별 수집하는 것이다. 전자의 방법은 중복 제거 시스템의 환경과 분석 환경을 동일하게 구성하기 위한 방법이다. 그러나 서버는 개인 PC와는 달리 대용량 데이터를 관리하기 때문에 시스템 전체를 디스크로 복제하는 것은 적절하지 않다. 후자의 방법에서는 시간 정보, 파일 이름, 키워드 검색 등으로 수집 대상 파일을 선별한다. 그러나 중복 제거된 파일을 대상으로 파일 내부의 내용에 대해서 키워드 검색을 할 수 없기 때문에 수집 대상으로 선정할 수 없다.

수집 과정에서는 활성 상태일 경우와 비활성 상태일 경우를 고려해야 하는데, 두 가지 경우에 대한 수집 절차를 최대한 통일하고, 수집해야 할 데이터를 최소화하여 데이터 수집에 소요되는 시간을 단축할 필요가 있다. 본 절에서는 앞서 언급한 제한 사항을 개선하고, 서버 시스템의 특성을 고려하여 중복 제거 데이터를 수집하는 방법을 제시한다.

Fig.11.은 수집 과정을 순서도로 나타낸 것이다. 먼저 수집 대상이 활성 상태인지 확인한다. 비활성 상태라면 쓰기 방지 장치를 활용하여 수집 대상의 디지털 저장 장치를 마운트 한다. 이후 과정부터는 수집 대상의 전원 상태와 관련 없이 동일한 과정을 따른다.

윈도우 서버 2012에서 데이터 중복 제거 기능은 볼륨마다 개별적으로 활성화시킬 수 있다. 따라서 각 볼륨마다 데이터 중복 제거 기능이 적용되었는지 확인해야 한다. 'System Volume Information' 폴더 하위의 디렉터리 목록을 검사하여 'Dedup' 폴더가 존재하면, 해당 볼륨은 데이터 중복 제거 기능이 활성화된 것이다.

중복 제거 기능이 활성화된 볼륨이 식별되면, 해당 볼륨의 $MFT 파일과 청크 저장소인 'Dedup'

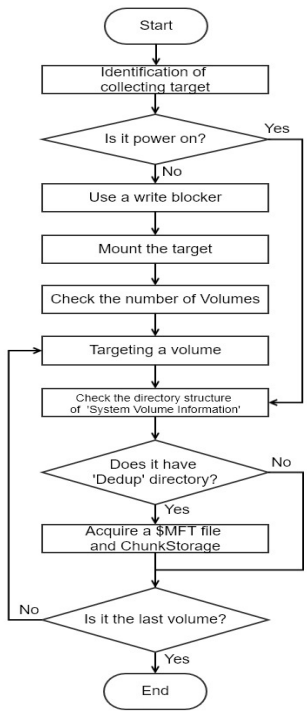

Fig. 11. Flow Chart of the Process to Collect Deduplicated Data

폴더를 수집한다. $MFT 파일과 'Dedup' 폴더만 수집하면, 분석 과정에서 원본 데이터를 복원하기 때문에 수집 과정에 소요되는 시간을 단축할 수 있다.

4.2 중복 제거 데이터 복원 방법

본 절에서는 중복 제거 파일의 원본 데이터를 복원하는 전체 과정에 대한 알고리즘을 Fig.12.로 제시한다. 수집 과정에서 수집한 $MFT와 'Dedup' 폴더가 복원 과정의 입력 데이터이다. 제시한 복원 알고리즘에 의해서 출력되는 데이터는 중복 제거된 파일 목록과 복원된 파일이다. 중복 제거 파일 목록은 입력받은 $MFT 파일을 분석하여 생성한다. 저장되는 내용은 파일 이름, 확장명, 파일 크기, 생성 시간, 수정 시간, 접근 시간 등으로 중복 제거 파일

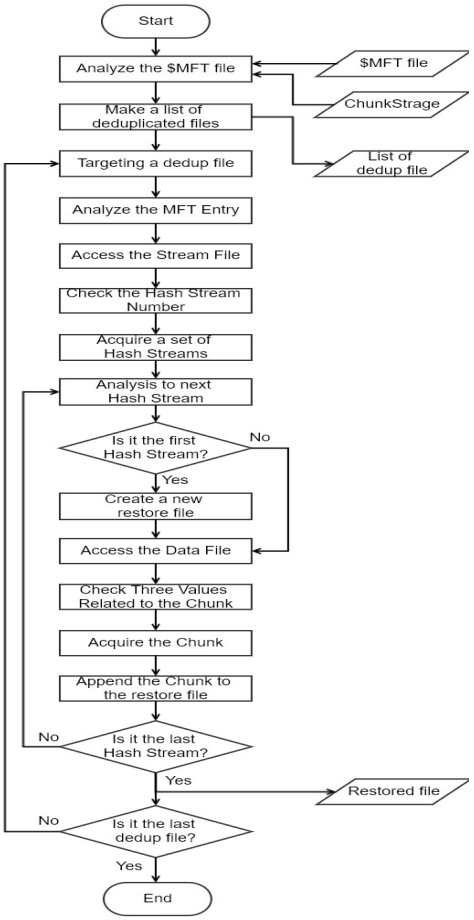

Fig. 12. Flow Chart of the Process to Restore Deduplicated File

의 메타데이터 정보를 보존하기 위한 목적으로 생성된다.

생성된 중복 제거 파일 목록을 바탕으로 중복 제거 파일마다 MFT Entry를 분석한다. 해당 과정은 청크 저장소의 식별자와 스트림 파일이 저장된 위치 정보를 파악하는 과정이다. 다음으로 스트림 파일에 접근해서 복원하려는 파일의 해시 스트림 전체를 획득해야하는데, 그 전에 MFT Entry에 저장된 해시 스트림 번호와 스트림 파일에 저장된 해시 스트림 번호가 동일한지 비교하여 정상적으로 복원 과정이 진행 중인지 확인한다.

다음은 획득한 해시 스트림 구조체를 개별적으로 분석하여 각 해시값과 연관된 데이터 파일이 저장된 위치 정보를 파악하는데, 첫 번째 해시 스트림 구조체를 분석하고 있다면 비어있는 새로운 파일을 생성한다. 새롭게 생성한 파일은 복원 과정이 완료되었을 때 결과적으로 중복 제거 파일을 복원한 파일이 된다.

분석한 데이터 파일의 위치정보를 바탕으로 데이터 파일에 접근하면 청크 데이터를 획득하기 전에 해시 스트림 구조체의 청크 번호, 청크 크기, 청크 해시값과 데이터 파일의 'Ckhr' 섹션 헤더에 저장된 청크 번호, 청크 크기, 청크 해시값을 비교하여 획득하려는 청크 데이터가 원본 데이터와 동일한지 확인한다. 세 값이 모두 일치하면 청크 데이터를 획득하고, 획득한 청크 데이터를 생성한 파일에 덧붙여서 기록한다. 만약 세 값이 일치하지 않다면 데이터 수집 과정에서 원본 데이터가 훼손되었거나 잘못된 'Dedup' 폴더가 입력된 것을 의미한다.

청크 데이터를 획득하는 반복 과정을 거쳐서 모든 청크 데이터를 재조합하면 중복 제거 파일 복원되고, 중복 제거 파일로 식별된 모든 파일에 대해서 청크 데이터 재조합 과정을 반복하면 중복 제거 파일이 모두 복원된다.

4.3 중복 제거 데이터 복원 과정에 대한 무결성 검증

제시한 중복 제거 파일 복원 방법을 디지털 포렌식 조사에 활용하려면 데이터 복원이 진행되는 도중에 원본 데이터가 변조되지 않고 정상적으로 복원되어 조사가 가능한 상태로 가시화된다는 것이 보장되어야 한다. 이를 검증하기 위하여 본 절에서는 4.1절과 4.2절에서 제시한 중복 제거 데이터 수집 및 복원 방법으로 임의의 중복 제거 파일을 복원했다.

Fig.13.은 복원하려는 중복 제거 파일의 해시 스

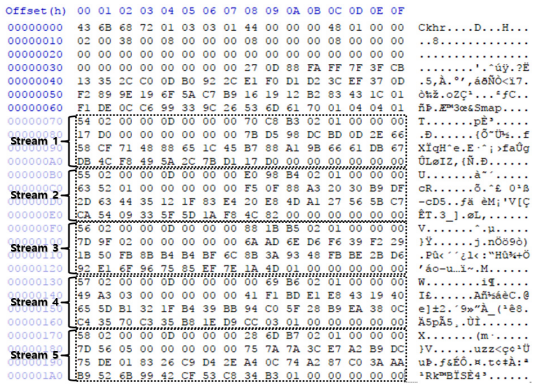

Fig. 13. The Hash Stream Example of a Deduplicated file

트림이다. 총 5개의 해시 스트림 구조체로 구성되어 있다는 것은 원복 파일을 복원하기 위하여 총 5개의 청크 데이터가 재조합되어야 한다는 것을 의미한다.

Table 1.은 복원 과정에서 재조합되기 위해서 필요한 청크 데이터와 원본 파일에서 재조합 순서가 일치하는 데이터의 MD5 해시값을 비교한 결과이다. 각 해시 스트림 구조체가 가리키는 청크 데이터는 원본 데이터의 각 부분으로 구성되는 바이너리 값과 동일한 데이터임을 알 수 있다. Table 2.는 원본 파일과 Fig. 12.의 복원 과정에 의해 출력된 파일의 해시값을 비교한 결과이며, 본 논문에서 제시한 방법이 중복 제거된 데이터를 이용하여 원본 파일을 복원할 수 있다는 사실을 확인할 수 있다.

Table 1. MD5 Chunk Hash Value Comparison Between Original and Restored

	Original	Restored
Chunk 1 (size: 0xD017)	41146331c5086fe5 e40ab6c83e468d4f	41146331c5086fe5 e40ab6c83e468d4f
Chunk 2 (size: 0x824C)	aa06790d74a1b640 dab40559e6b647d4	aa06790d74a1b640 dab40559e6b647d4
Chunk 3 (size: 0x14D1A)	a8690dfc0328c21f 1d86d383dd9af12a	a8690dfc0328c21f 1d86d383dd9af12a
Chunk 4 (size: 0x103CC)	ad7a5d92bd3ed928 5002fe4c9d1b6a69	ad7a5d92bd3ed928 5002fe4c9d1b6a69
Chunk 5 (size: 0x1B334)	fd1de2934f082a0c f76e5df2a836a2dc	fd1de2934f082a0c f76e5df2a836a2dc

Table 2. Hash Value Comparison

	MD5	SHA1
Original File	2235165855cb114b 7094e62b5426cae5	f1f14ce19a2e20cc457c a3ab257f80aab7826f1c
Restored File	2235165855cb114b 7094e62b5426cae5	f1f14ce19a2e20cc457c a3ab257f80aab7826f1c

V. 결 론

마이크로소프트는 가장 최근에 출시된 윈도우 10을 대상으로도 중복 제거 기능을 추가하는 사항에 대하여 검토 중이다[14]. 이를 통해 중복 제거 기능이 사용되는 범위가 더 확대될 것으로 예상되며, 디지털 포렌식 조사를 위해서도 추가적인 연구가 지속적으로 필요하다. 기존 연구에서는 단순히 중복 제거 기능이 활성화된 시스템을 대략적으로 분석하여 중복 제거 파일에 대한 복원 가능성을 보여주었으나 그 한계를 확인할 수 있었다.

따라서 본 논문에서는 중복 제거 기능이 적용된 마이크로소프트의 윈도우 서버 2012 시스템을 대상으로 중복 제거 기능에 의해서 생성된 청크 저장소의 구조를 분석하였다. 그리고 데이터 중복 제거 기능이 활성화된 볼륨의 식별 방법과 식별된 볼륨의 중복 제거 데이터를 수집하는 방법과 중복 제거된 파일을 복원 방법을 제시하였다. 윈도우 서버 2012에서는 다른 시스템과 달리 더 나은 중복 제거 기능을 제공하기 위하여 분할된 청크를 압축하는 경우가 있는데 압축 방식과 압축된 청크 데이터를 압축 해제하는 방법을 확인하였다. 제시한 방법을 활용한다면 대상 시스템의 저장매체를 복제하지 않더라도 항상 동일한 절차에 의해 파일 분석이 가능하다. 이러한 관점에서 본 논문의 연구결과는 디지털 포렌식 관점에서 중복 제거 데이터를 수집하고 분석하기 위한 기초 연구 자료로 활용할 수 있을 것이다.

References

[1] Gantz, John, and David Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," IDC iView: IDC Analyze the future 2007, 2012.

[2] Dario Lanterna and Antonio Barili,

"Forensic analysis of deduplicated file systems," Digital Investigation, vol. 20, pp. 99-106, Mar. 2017.

[3] Muthitacharoen, Athicha, Benjie Chen, and David Mazieres, "A low-bandwidth network file system," ACM SIGOPS Operating Systems Review, Vol. 35, No. 5, pp. 174-187, Oct. 2001.

[4] Min, Jaehong, Daeyoung Yoon, and Youjip Won, "Efficient deduplication techniques for modern backup operation," IEEE Transactions on Computers Vol. 60, No. 6, pp. 824-840, June. 2011.

[5] Harnik, Danny, Benny Pinkas, and Alexandra Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," IEEE Security & Privacy Vol. 8, No. 6, pp. 40-47, Dec. 2010.

[6] Carlton, Gregory H, and Joseph Matsumoto, "A survey of contemporary enterprise storage technologies from a digital forensics perspective," The Journal of Digital Forensics, Security and Law: JDFSL, Vol. 6, No. 3, pp. 63-74, 2011.

[7] Park, Jungheum, Hyunji Chung, and Sangjin Lee, "Forensic analysis techniques for fragmented flash memory pages in smartphones," Digital Investigation Vol. 9, No. 2, pp. 109-118, Nov. 2012.

[8] Neuner, Sebastian, et al, "Gradually improving the forensic process," Availability, reliability and security (ares), 2015 10th international conference on. IEEE, pp. 404-410, Aug. 2015.

[9] Neuner, Sebastian, Martin Schmiedecker, and Edgar Weippl, "Effectiveness of file based deduplication in digital forensics," Security and Communication Networks, Vol. 9, No. 15, pp. 2876-2885, Jan. 2016.

[10] Rabin, Michael O., "Fingerprinting by random polynomials," TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[11] El-Shimi, Ahmed, et al, "Primary Data Deduplication-Large Scale Study and System Design," USENIX Annual Technical Conference, Vol. 2012, pp. 285-296, June. 2012.

[12] Windows Dev Center, "Reparse Points," https://msdn.microsoft.com/ko-kr/library/windows/desktop/aa365503(v=vs.85).aspx

[13] Microsoft Developer Network, "[MS-XCA]: Xpress Compression Algorithm," https://msdn.microsoft.com/en-us/library/hh554002.aspx, Sept. 2017.

[14] Windows Server, "Add deduplication support to client OS," https://windowsserver.uservoice.com/forums/295056-storage/suggestions/9011008-add-deduplication-support-to-client-os, July. 2015.

〈저자소개〉

손 관 철 (Gwancheol Son) 학생회원
2016년 2월: 연세대학교 컴퓨터공학과 졸업
2016년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 디지털 포렌식, 역공학, 파일시스템

한 재 혁 (Jaehyeok Han) 학생회원
2011년 2월: 서울시립대학교 수학과 졸업
2016년 2월: 고려대학교 정보보호대학원 공학석사
2016년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 디지털 포렌식, 파일시스템, 데이터 마이닝

이 상 진 (Sangjin Lee) 종신회원
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, 심층암호, 해쉬함수