KIPS Tr. Comp. and Comm. Sys.
Vol.6, No.2 pp.95~104 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2017.6.2.95

Digital Forensic Investigation of HBase

Aran Park† · Doowon Jeong†† · Sang Jin Lee†††

ABSTRACT

As the technology in smart device is growing and Social Network Services(SNS) are becoming more common, the data which is difficult to be processed by existing RDBMS are increasing. As a result of this, NoSQL databases are getting popular as an alternative for processing massive and unstructured data generated in real time. The demand for the technique of digital investigation of NoSQL databases is increasing as the businesses introducing NoSQL database in their system are increasing, although the technique of digital investigation of databases has been researched centered on RDMBS. New techniques of digital forensic investigation are needed as NoSQL Database has no schema to normalize and the storage method differs depending on the type of database and operation environment. Research on document-based database of NoSQL has been done but it is not applicable as itself to other types of NoSQL Database. Therefore, the way of operation and data model, grasp of operation environment, collection and analysis of artifacts and recovery technique of deleted data in HBase which is a NoSQL column-based database are presented in this paper. Also the proposed technique of digital forensic investigation to HBase is verified by an experimental scenario.

Keywords : HBase, NoSQL, Database Forensic, Digital Forensic, Hadoop

HBase에 대한 디지털 포렌식 조사 기법 연구

박 아 란† · 정 두 원†† · 이 상 진†††

요 약

최근 스마트 기기의 발전과 소셜 네트워크 서비스(SNS)의 대중화로 기존 관계형 데이터베이스(RDBMS)에서는 처리하기 어려운 데이터들이 증가하고 있다. 이러한 대용량의 비정형 데이터를 실시간으로 처리하기 위한 대안으로 비관계형 데이터베이스(NoSQL DBMS)가 각광받고 있다. 데이터베이스 디지털 포렌식 조사 기법은 대부분 관계형 데이터베이스를 대상으로 연구되어왔으나, 최근 NoSQL DBMS를 도입하는 기업이 증가하면서 NoSQL DBMS에 대한 디지털 포렌식 기법의 수요도 증가하고 있다. NoSQL DBMS는 정규화할 스키마가 존재하지 않고, 데이터베이스 종류나 운영환경에 따라 저장방식이 상이하기 때문에 디지털 포렌식 조사 시 이를 고려한 새로운 기법들이 필요하다. NoSQL DBMS 중 문서형 데이터베이스에 대한 연구는 진행되어 왔지만, 이를 다른 종류의 NoSQL DBMS에 그대로 적용하기엔 한계가 있다. 이에 본 논문에서는 NoSQL DBMS 중 컬럼형 데이터베이스인 HBase의 구동 방식과 데이터 모델을 소개하고, 운영환경 파악과 아티팩트 수집 및 분석, 삭제된 데이터의 복구 방안에 대해 제안하여 이를 바탕으로 HBase에 대한 디지털 포렌식 조사 기법에 대해 연구하였다. 또한 실험 시나리오를 통해 제안된 HBase에 대한 디지털 포렌식 조사 기법을 검증한다.

키워드 : HBase, NoSQL, 데이터베이스 포렌식, 디지털 포렌식, 하둡

1. 서 론

최근 스마트폰과 같은 휴대용 기기의 발전과 다양한 애플리케이션들의 개발로 인해 소셜 네트워크 서비스(SNS)나 인터넷을 통한 정보 공유가 활발해지고 있다. 데이터의 유형이 다양해지고 기존의 관계형 데이터베이스(RDBMS)와 같이 데이터를 정형화하여 저장하는 것이 어려워지고 있다. 이를 처리하기 위한 방안으로 비관계형 데이터베이스인 NoSQL DBMS가 주목받고 있다. NoSQL DBMS는 데이터를 고정된 스키마로 정규화하지 않아도 되고, 대용량의 데이터를 물리적으로 나뉜 여러 서버를 이용하여 관리하므로 많은 양의 데이터를 실시간으로 빠르게 처리할 수 있다[1, 2]. 이러한 NoSQL DBMS의 장점을 이용해 최근 기업이나 기관 또는 다양한 서비스에서 NoSQL DBMS를 사용하는 빈도수가 높아지고 있다[3].

본 논문에서는 이러한 NoSQL DMBS 중 페이스북(Facebook),

※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단-공공복지안전사업의 지원을 받아 수행된 연구임(2012M3A2A1051106).
† 준 회 원 : 고려대학교 정보보호대학원 정보보호학과 석사과정
†† 비 회 원 : 고려대학교 정보보호대학원 정보보호학과 석·박사통합과정
††† 종신회원 : 고려대학교 정보보호대학원 교수
Manuscript Received : November 1, 2016
Accepted : November 22, 2016
* Corresponding Author : Sang Jin Lee(sangjin@korea.ac.kr)

카카오톡(Kakaotalk), 라인(Line), 비트윈(Between)과 같은 SNS에서 사용되는 HBase에 대한 디지털 포렌식 조사 기법을 제안한다. 또한 HBase의 운영 방법 및 데이터 저장 방식에 대한 연구와 디지털 포렌식 수사 시 수집해야 할 아티팩트들과 이를 분석하는 방법 그리고 삭제된 데이터를 복구하는 방법을 제안한다.

2. 관련 연구 및 배경지식

2.1 관련 연구

기업이나 기관에서는 데이터의 무결성, 보안, 권한 등을 보장할 수 있고, 모델링이 간편한 관계형 데이터베이스(RDBMS)를 사용해왔다. 이러한 이유로 관계형 데이터베이스(RDBMS)에 관한 디지털 포렌식 연구는 활발하게 진행되어 왔다.

Khanuja 등은 관계형 데이터베이스인 Mysql의 내부구조와 아티팩트들을 분석하여, 디지털 포렌식 조사 프레임 워크를 제안하였다[4]. Aldhaqm 등은 데이터베이스에 대한 포렌식 수사 시 아티팩트, 분석도구, 분석방법 등의 연구들을 바탕으로 데이터베이스의 디지털 포렌식 알고리즘을 제안하였다[5]. Pavlou 등은 RDBMS에서 데이터 조작 시 이를 탐지하는 기법에 대해 연구하였다[6]. 또한 Jong-Hyun Choi는 Oracle 데이터베이스에서 삭제된 레코드 복구 방법을 제안하고, 이를 바탕으로 도구를 개발하였다[7]. Adedayo 등은 많이 사용되는 6가지 RDBMS를 대상으로 데이터베이스 포렌식에서 중요한 역할을 하는 로그파일을 이용해 데이터베이스 재구성에 관한 연구를 했다[8].

RDBMS의 경우 스키마 기준으로 정규화하여 데이터를 관리하는 반면, NoSQL DBMS의 경우 정규화할 스키마가 존재하지 않기 때문에 디지털 포렌식 조사 시 이를 고려한 새로운 기법들이 필요하다.

NoSQL DBMS와 관련된 연구는 Jong-Seong Yoon 등이 MongoDB에 대한 디지털 포렌식 조사 절차와 아티팩트 수집 방법 및 세부 조사 방법에 대해 연구했고[9], Jae Mun Choi 등이 Redis의 소개와 디지털 포렌식 관점에서 확인해야 할 아티팩트들과 이를 분석하는 방법 그리고 삭제된 데이터 복구 기법에 대하여 연구했다[10].

MongoDB는 Document형 데이터베이스로 각 레코드들을 JSON의 형태로 저장하고, Redis는 Key-Value형 데이터베이스로 데이터를 Key와 Value의 쌍으로 저장한다. 반면, HBase는 컬럼형 데이터베이스로 각 컬럼들을 컬럼패밀리(Column-Family)로 관리하기 때문에 데이터의 저장방식과 구조가 MongoDB나 Redis와는 상이하다. 또한 데이터 추출하는 방법에도 차이가 있다. MongoDB의 경우 DB 파일을 직접 추출하거나 명령어를 통해 데이터를 추출할 수 있고, Redis는 메모리 기반의 데이터베이스이므로 메모리 위에 존재하는 데이터를 하드디스크에 파일로 추출해야 한다. HBase는 주로 하둡 분산 파일시스템(HDFS) 위에서 동작하므로 HDFS에 저장된 파일을 로컬 시스템으로 추출한 후 해당 파일을 획득할 수 있다. 이와 같이 MongoDB, Redis와의 구조 및 운영 특성의 차이로 선행연구를 그대로 HBase에 적용하여 디지털 포렌식 조사를 수행하는 데에는 한계가 있다.

HBase에 대한 연구로는 Google에서 HBase의 모델인 BigTable에 대한 내용을 논문의 형태로 발표했다[11]. Xiaoming Gao 등은 HFile의 구조를 간단히 분석했다[12]. 하지만 HBase를 디지털 포렌식 관점으로 접근한 연구나 HFile의 자세한 구조, 삭제된 데이터의 복구 방안에 대한 연구는 진행되지 않았다.

2.2 배경지식

HBase는 Google의 BigTable을 모델로 한 컬럼형 데이터베이스로 2007년 Michael Stack과 Jim Kellerman에 의해 하둡(Hadoop) 프로젝트의 일부로 시작된 오픈소스 소프트웨어이며, 현재는 Apache 프로젝트로 개발이 진행되고 있다[13].

HBase는 Hadoop의 기능 중 하나인 맵리듀스(Map Reduce)를 이용해 대량의 데이터를 병렬로 처리하여 데이터를 분산처리할 수 있고, 높은 확장성을 가진다[13]. 또한 분산 시스템을 통제하는 Master와 복제된 Slave들로 구성되어 데이터의 일관성이 보장되므로 데이터의 실시간 접근이나 특정 셀에 접근 등이 가능하다. 이러한 이유로 페이스북(Facebook), 카카오톡(Kakaotalk), 라인(Line), 비트윈(Between)과 같이 대용량 데이터의 실시간 처리가 필요한 SNS에서 HBase가 사용된다[14]. 대부분의 서비스에서 주요한 데이터는 RDBMS에 저장하고, 로그나 설정파일 등 부수적인 데이터를 NoSQL에 저장하지만, 실시간으로 메시지를 처리하는 데에도 HBase가 사용된다.

HBase는 하둡 명령어를 이용해 파일에 접근 및 추출이 가능하다[15]. 또한 하둡의 다양한 기능들과 함께 사용할 수 있어 대용량 데이터의 분산처리에 효율적으로 사용할 수 있다. HBase는 HBase Shell에서 명령어를 통해 데이터의 삽입, 수정, 삭제가 가능하고, 테이블 정보나 상태를 확인할 수 있다.

본 논문에서는 HBase의 동작 및 운영 환경과 데이터 저장 방식에 대한 분석과 삭제된 데이터 복구 방안을 연구하여 HBase에 대한 디지털 포렌식 조사 절차 및 기법을 제안하고자 한다. 3장에서는 기존의 데이터베이스 포렌식 방법을 기반으로 HBase에 대한 새로운 알고리즘의 필요성을 제안한다. 4장에서는 HBase의 동작 환경과 데이터 저장 방식에 대한 분석을 진행한다. 5장에서는 이를 바탕으로 HBase에 대한 디지털 포렌식 조사를 위한 절차와 데이터가 저장되는 파일의 구조 및 분석 방법에 대해 설명하고, 삭제된 데이터의 복구기법을 제시한 후 HBase에 대한 디지털 포렌식 조사 기법에 대해 제안한다. 6장에서는 실험을 통해 5장에서 제시한 디지털 포렌식 조사 기법을 검증한다.

3. 데이터베이스 포렌식 절차

3.1 RDBMS에 대한 디지털 포렌식 절차

일반적인 RDBMS에 대한 디지털 포렌식 절차는 Fig. 1과 같다[16]. RDBMS에는 정규화된 스키마와 테이블간의 관계

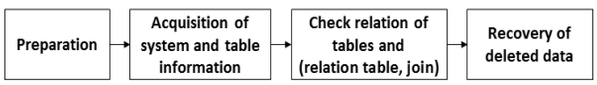

Fig. 1. The General Process of Database Forensic

가 존재한다. 따라서 테이블의 스키마 정보를 획득하고, 테이블 간의 관계를 파악해야 한다. 또한 분석한 스키마 정보를 기반으로 데이터 파일의 미할당 영역 내에 존재하는 삭제된 데이터를 복구할 수 있다.

3.2 NoSQL DBMS에 대한 디지털 포렌식 절차

NoSQL DBMS는 고정된 스키마가 존재하지 않으므로 테이블의 스키마 정보와 관계 파악이 불필요하다. 반면 대용량 데이터를 관리하는 NoSQL DBMS는 주로 분산된 환경에서 사용되므로 이에 대한 조사 및 분석이 추가로 필요하다. 또한 NoSQL DBMS는 제품군 별로 데이터 저장방식과 운영환경이 다르므로 디지털 포렌식 조사 절차 수립 시 이를 고려해야 한다[1].

NoSQL DBMS는 문서형 데이터베이스, 컬럼형 데이터베이스, Key-Value 데이터베이스, 그래프 데이터베이스로 분류할 수 있다[17]. Jong-Seong Yoon 등은 이러한 NoSQL 중 문서형 데이터베이스에 대한 디지털 포렌식 조사 단계를 '준비-논리적 증거 획득-분산된 증거 식별 및 획득-분석-보고서 작성'의 5가지로 나누었다[18].

문서형 데이터베이스의 경우 데이터를 XML이나 JSON의 형태로 저장하는 반면 HBase와 같은 컬럼형 데이터베이스는 컬럼패밀리를 이용해 데이터를 관리한다. 따라서 '논리적 증거 획득' 단계에서 문서형 데이터베이스에 대한 디지털 포렌식 기법을 그대로 적용하기엔 한계가 존재하므로, 조사 기법에 컬럼패밀리에 대한 식별 및 수집 단계가 추가 되어야 한다. 또한 '분산된 증거 식별 및 획득' 단계에서 HBase가 하둡 분산 파일시스템에서 동작하는 경우 데이터가 분산 저장되어 있어 HBase의 특성을 고려한 증거 식별 및 획득, 선별 수집에 대한 절차 수립이 필요하다. 데이터 저장 방식과 형태 또한 기존에 연구된 DBMS들과 상이하므로 이를 고려한 새로운 삭제된 데이터 복구 알고리즘을 개발해야 한다. Table 1은 RDBMS와 NoSQL DBMS의 특징을 비교한 것이다.

따라서 본 논문에서는 기존에 연구된 NoSQL 문서형 데이터베이스에 대한 디지털 포렌식 조사 기법을 기반으로 컬럼형 데이터베이스인 HBase에 대한 디지털 포렌식 조사 기법을 제안하고자 한다.

Table 1. Comparison of RDBMS and NoSQL DBMS

	RDBMS (Oracle, MySQL etc.)	NoSQL DBMS	
		Document Database (MongoDB etc.)	Column Store Database (HBase etc.)
Schema	O	X	X
Feature	Depends on the database	XML, JSON	Column Family

4. HBase 동작 환경 및 데이터 저장 방식

4.1 데이터베이스 구성(구축) 및 운영

HBase의 데이터 저장 방식에는 Stand-alone 모드와 Distributed 모드가 있다. Stand-alone 모드는 HBase 서버 하나만 사용하는 경우로 로컬 파일시스템을 사용한다. 이 경우 HBase 데몬과 Zookeeper는 모두 하나의 JVM (Java Virtual Machine) 상에서 동작한다.

Distributed 모드는 Pseudo-distributed 모드와 Fully-distributed 모드가 있으며, 두 경우 모두 하둡(Hadoop)을 파일시스템으로 사용하여 쉽게 분산구성이 가능하다. 먼저, Pseudo-distributed 모드는 물리적으로 하나의 서버 위에 여러 HBase가 사용되는 경우로 각각의 데몬이 하나의 노드에서 독립적으로 동작한다. 실제로 사용되는 환경은 아니지만 HBase 관련 소스코드 분석 및 테스트와 버그 수정 등을 위한 테스트 환경으로 사용된다.

Fully-distributed 모드는 HBase가 물리적으로 분리된 여러 서버에 나뉘어 동작하는 경우로 하나의 데이터가 각각의 분산된 서버에 나뉘어 저장된다. 이 모드는 실제 기업에서 사용되는 환경으로 각각의 데몬도 분산되어 동작한다.

분산 구성 시 HBase는 Master와 Slave로 나뉘어 동작한다. 이때, Master는 HMaster를 의미하고 Slave는 Region Server를 의미한다. HMaster는 Region들을 Region Server에 분배하고, 이러한 Region Server들을 관리하는 역할을 하고, 실제 데이터는 Region Server에 저장된다. 로우들(Row, 행)의 집합인 Region은 분산을 이루는 기본 단위로 각각의 Region Server는 여러 개의 Region들을 포함하고 있다.

4.2 데이터 저장 방식

HBase는 메모리에 데이터를 저장하여 빠른 속도로 읽고 쓰는 것이 가능하다. HBase에서의 데이터 저장 방식은 Fig. 2와 같다. Client는 PUT, DELETE 등의 명령어로 데이터를 추가/수정/삭제 할 수 있고, 그 결과는 HFile에 저장된다. 데이터가 업데이트 될 때마다 Region Server는 데이터를 메모리에 존재하는 데이터베이스인 Memstore에 저장한다. Memstore에 Write한 데이터 크기가 'hbase.hregion.memstore.flush.size' 속성에 지정한 Memstore의 크기를 초과하거나, Memstore 크기의 합이 'hbase.regionserver.global.memstore.upperLimit/lowerLimit' 속성에 지정한 값에 도달하면, Memstore에 저장된 데이터들이 디스크(HFile)에 저장(Flush)된다. 이러한 Memstore와 HFile은 컬럼패밀리 당 하나씩 존재한다.

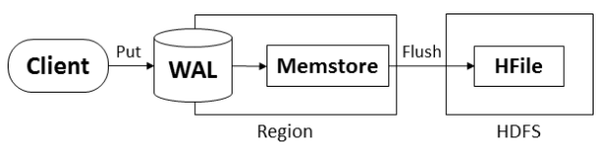

Fig. 2. Procedure of Storing Data from HBase

Table 2. The Interpretation of the Example Table, 'human'

RowKey	Column-Family	Column-Qualifier	Timestamp	Column-Value
row01	address	apartment	1467190609624	okApt
		street	1467190543289	sungbuk
	user	email	1467190486003	colorBora@abc.com
		name	1467190435053	bora
row02	user	name	1467190752419	jini
row03	user	email	1467190878291	sweetGuul@orange.com
		name	1467190790445	guri

메모리에 저장되는 데이터는 휘발성이므로 시스템에서 전원이 꺼지거나 오류 동작으로 인해 장애 발생 시 데이터가 유실·손상될 수 있다[20]. 따라서 추가·수정된 데이터는 Memstore에 저장되기 전 WAL(Write-Ahead-Log)에 저장된다.

4.3 데이터 모델

HBase는 기본적으로 Key와 Value의 쌍으로 이루어진 데이터를 컬럼 단위로 저장한다(Fig. 3). 하나 이상의 컬럼들이 모여 하나의 로우(Row)를 구성하며, 테이블은 로우들의 모음으로 구성된다. 각각의 로우는 로우 키(Row Key)를 갖고, 이는 로우를 식별할 때 사용된다[19].

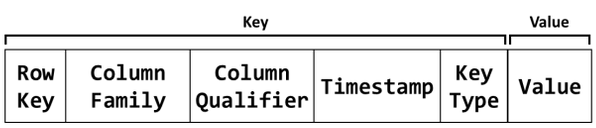

Fig. 3. Column Structure (Key-Value)

로우의 각 컬럼은 '컬럼패밀리(Column-Family): 퀄리파이어(Qualifier)' 형태로 그룹화 되고, 이를 이용해 데이터의 컬럼들을 관리한다. 또한 모든 컬럼들에는 시간을 나타내는 타임스탬프(Timestamp)가 시스템에서 자동으로 생성된다. HBase는 로우 키와 타임스탬프, 컬럼 키로 색인이 가능하고, 각각의 값(셀)은 로우 키를 기준으로 사전식으로 정렬되며, 타임스탬프에 대해 내림차순으로 저장된다.

Fig. 4는 로우와 컬럼 구조를 설명하기 위해 예시로 생성한 'human' 테이블이다. 'human' 테이블을 로우·컬럼 단위로 해석한 표는 Table 2와 같다.

5. HBase에 대한 디지털 포렌식 조사 기법

HBase는 운영되는 환경이 설정 방식에 따라 다르기 때문에 디지털 포렌식 조사 시 먼저 운영 환경을 파악해야 한다. 그러므로 운영 환경에 필요한 설정 파일들의 수집과 명령어를 통한 동작 환경 파악을 한다. 이후 HDFS 사용 여부와 분산 여부, 분산 모드를 확인한다. HDFS를 사용하고, 분산된 경우 Master서버의 설정 파일들을 통해 해당 노드에서 동작하는 Region Servers(Slaves)의 정보를 확인해야 한다.

기존에 제안된 NoSQL 문서형 데이터베이스에 대한 디지털 포렌식 조사 기법 중 '논리적 증거 획득' 단계에서 컬럼 패밀리의 식별 및 수집 단계와 데이터에 대한 선별 수집 단계가 추가되어야 한다. 컬럼패밀리의 식별은 HBase의 필터 기능을 이용해 확인하고, 이를 통해 실제 데이터가 저장되는 HFile과 WAL 파일을 선별 수집한다.

또한 '분산된 증거 식별 및 획득' 단계에서 HBase가 HDFS를 사용하는 경우 Region Servers에 대한 정보를 통한 증거 획득이 필요하기 때문에 Region Servers의 위치 파악 단계를 추가해야 한다.

삭제된 데이터의 복구가 필요한 경우 HBase의 데이터 저장방식 특징을 이용하여 본 논문에서 연구한 복구 알고리즘을 통해 데이터를 복구한다.

이를 바탕으로 Fig. 5와 같이 HBase에 대한 디지털 포렌식 조사 기법을 제안한다.

```
hbase(main):003:0> scan 'human'
ROW                                   COLUMN+CELL
 row01                                column=address:apartment, timestamp=1467190609624, value=okApt
 row01                                column=address:street, timestamp=1467190543289, value=sungbuk
 row01                                column=user:email, timestamp=1467190486003, value=colorBora@abc.com
 row01                                column=user:name, timestamp=1467190435053, value=bora
 row02                                column=user:name, timestamp=1467190752419, value=jini
 row03                                column=user:email, timestamp=1467190878291, value=sweetGuul@orange.com
 row03                                column=user:name, timestamp=1467190790445, value=guri
3 row(s) in 0.4120 seconds
```

Fig. 4. The Example Table, 'human'

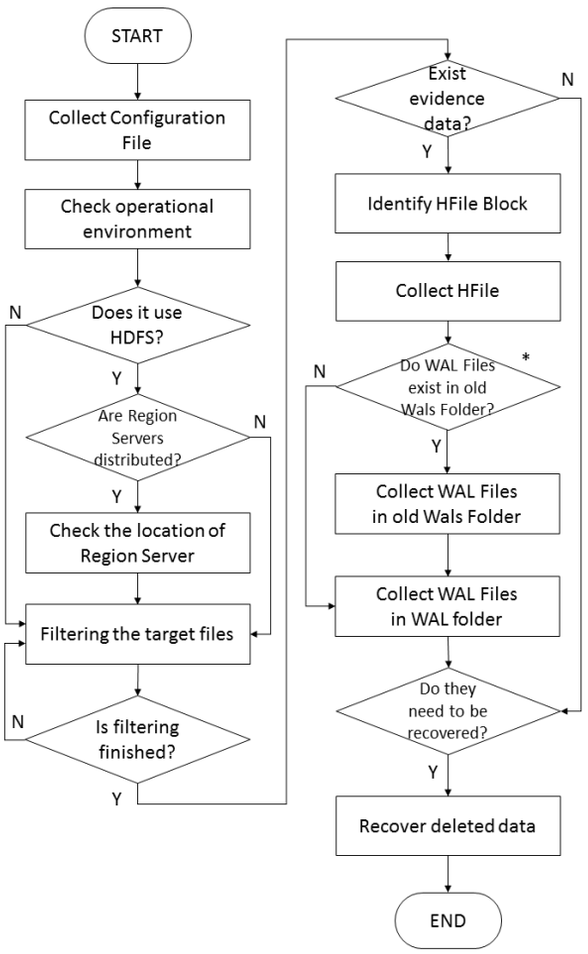

Fig. 5. The Procedure for HBase Forensic Investigation

5.1 운영환경 파악

NoSQL을 대상으로 디지털 포렌식 조사 시 분석 대상의 데이터가 대용량이고, 물리적으로 분리된 서버에 나뉘어 저장될 수 있음을 고려해야 한다. 또한 환경 설정에 따라 분석 대상이 달라질 수 있으므로 조사 대상 파일 선별 전 서버의 설정 환경도 고려해야 한다. HBase가 HDFS 위에서 동작하는 경우 HBase와 Hadoop의 환경설정 모두 확인해야 데이터를 추출할 수 있다.

1) 설정 파일 수집 및 기본 정보 분석

현재 동작하는 Region Server의 개수, HBase의 버전 등의 기본정보는 HBase Shell에서 명령어 'status', 'version'으로 확인할 수 있다.

데이터가 저장될 경로, Zookeeper가 실행될 노드, HBase 분산 여부 등의 기본 정보는 'hbase-site.xml' 파일에 존재하므로 해당 파일을 추출해야 한다. Table 3은 'hbase-site.xml' 파일에서 사용되는 속성들과 그 의미를 나타낸 것이며, 해당 파일은 HBase 설치 경로에서 획득할 수 있다.

Table 3. Attribute and Description of 'hbase-site.xml'

Attribute	Description
hbase.rootdir	The path where data is sotred, distributed mode
hbase.master	Setting a master port
hbase.zookeeper.quorum	Nodes for execute the Zookeeper
hbase.zookeeper.property.dataDir	Path of the Zookeeper
hbase.cluster.distributed	Status of the distribution

2) HDFS와 HBase 사용 여부

Table 4는 HDFS와 HBase의 사용 여부를 확인하는 방법이다. HBase에서 HDFS 사용 여부는 명령어와 설정파일을 이용해 확인할 수 있다.

먼저 명령어 'jps'는 현재 서버에서 동작 중인 데몬을 보여주는 것으로 HDFS의 동작여부와 HBase 서버의 역할(HMaster 또는 HRegionServer)을 확인할 수 있다.

또한 Table 3에서와 같이 설정파일 'hbase-site.xml' 파일의 속성 중 'hbase.rootdir'는 HBase의 데이터가 저장되는 경로를 나타내는 값으로 해당 값이 'hdfs://'인 경우 HDFS를 사용하고, 'file://'인 경우 로컬파일시스템을 사용하는 것이다.

Table 4. Condition of HDFS & HBase Operating

	Method	operating	not operating
HDFS	Check configuration file	'hdfs://~'	'file://~'
	Check deamon	NameNode or Secondary or DataNode	※ If any deamon does not exist, it will not
HBase	Check deamon	HMaster or HRegionServer	

3) 분산 여부 확인

HBase는 데이터를 물리적으로 다른 서버에 나누어 저장할 수 있으므로 분산 여부를 확인하는 것이 중요하다. Table 5는 분산 여부를 확인하는 방법이다. 분산여부는 'hbase-site.xml' 파일과 'regionservers' 파일에서 확인할 수 있다.

Table 3에서와 같이 'hbase-site.xml' 파일의 속성 중 'hbase.cluster.distributed'는 분산 여부를 나타내는 속성으로 해당 값이 'false'인 경우 Stand-alone 모드이고, 'true'인 경우 Distributed 모드이다.

Distributed 모드인 경우 'regionservers' 파일을 통해 Region Server의 정보를 확인할 수 있고, 분산된 서버의 IP와 Zookeeper 이름 등을 확인할 수 있다.

Table 5. Check State of Distributed

	Method	operating
Stand-alone	Check distributed attribute	false
Distributed	Check distributed attribute	true (Check 'regionservers' file)

5.2 데이터 수집 및 분석

1) 데이터 필터링

운영환경 파악 후 수집하고자 하는 데이터가 저장된 Region Server와 Block을 식별해야 한다.

HBase는 방대한 양의 데이터를 다수의 서버에 저장하고 관리하기 때문에 사건과 관련된 데이터를 추출할 때, 모든 HFile을 확인하는 것은 비효율적이다. 따라서 해당 사건과 관련 있는 데이터를 선별하여 수집하는 것이 중요하다.

HBase Shell에서 'Filter' 기능을 이용해 데이터의 선별 수집이 가능하다. Filter는 RDBMS에서의 쿼리와 유사한 역할을 하며, Key나 Value를 이용해 키워드가 포함된 로우를 검색하여 데이터의 유무를 확인할 수 있다. Filter의 종류는 명령어 'show_filters'로 확인할 수 있고, Table 6은 Filter의 일부를 나타낸 것이다. 여러 Filter들을 'AND'와 'OR'로 조합하여 세밀한 검색이 가능하고, 사용자가 원하는 Filter를 직접 구현할 수 있다.

Table 6. Part of 'show_filters'

Filter Name	Description
PrefixFilter	Search for data using a prefix
TimestampsFilter	Search for data using a specific timestamp
ValueFilter	Search for data using a specific value
ColumnPrefixFilter	Search for data using a specific column value
QualifierFilter	Search for data using a specific qualifier value
FamilyFilter	Search for data using a specific column-family

필터링 작업으로 획득한 로우의 컬럼패밀리를 이용해 데이터가 저장된 HFile을 추출한다. 조사 대상의 데이터가 존재하는 컬럼패밀리 폴더에서 HFile을 추출할 수 있으며, HBase에 저장된 데이터는 분산 모드에 관계없이 HFile과 WAL 파일에서 획득할 수 있다. 수집해야 할 데이터의 위치를 파악한 후 Stand-alone 모드의 경우 로컬시스템에서 HFile을 추출하고, Distributed 모드의 경우 Hadoop 명령어 '-copyToLocal'을 이용해 HDFS에 존재하는 파일을 로컬시스템으로 추출한다.

2) HFile

HBase는 데이터 삽입, 수정, 삭제 시 해당 데이터를 컬럼패밀리 단위로 HFile에 최종적으로 저장한다. 이처럼 HFile에는 HBase에 존재하는 실제 데이터가 저장되므로 해당 파일을 수집하는 것은 HBase에 존재하는 데이터를 수집하는 것을 의미한다.

HFile이 저장되는 곳은 'hbase-site.xml' 파일의 'hbase.rootdir' 속성에서 알 수 있으며, '/{hbase-root}/data/default/{Table_Name}/{인코딩된_리전명}/{Column-Family_Name}'에 HFile들이 존재한다.

HFile에서 데이터가 존재하는 영역의 구조는 Fig. 6과 같다. Fig. 7은 하나의 Row Header와 Row Data 영역을 나타낸 것이다. HFile은 Key-Value의 형태로 데이터를 저장하고, 각각의 컬럼들은 타임스탬프를 가진다.

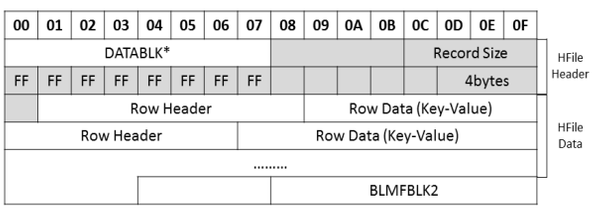

Fig. 6. Structure of HFile (1)

3) WAL (HLog)

다른 RDBMS와 유사하게 HBase에서도 트랜젝션을 기록하는 WAL 파일이 존재한다[8, 21].

WAL(Write Ahead Log)은 메모리에 입력되기 전 기록되는 파일로 Region Server 당 하나씩 존재하며 HLog라고도 부른다. 실제로 PUT이나 DELETE와 같이 스토어(Store)에 발생하는 모든 이벤트들은 WAL의 Edit Block에 기록된다. 따라서 Region Server 장애 시 Edit Block들을 이용해 데이터를 복구할 수 있다. 디지털 포렌식 조사 시 서버의 오류나 장애로 HFile에 저장되지 못한 데이터를 수집하기 위해 WAL 파일을 추출해야 한다. WAL 파일이 저장되는 경로는 '/{hbase-root}/WALs/slave명,PORT,시간값/'이며, 해당 폴더에 'meta' 파일과 'default' 파일이 존재한다. Fig. 8은 WAL 파일의 구조이다.

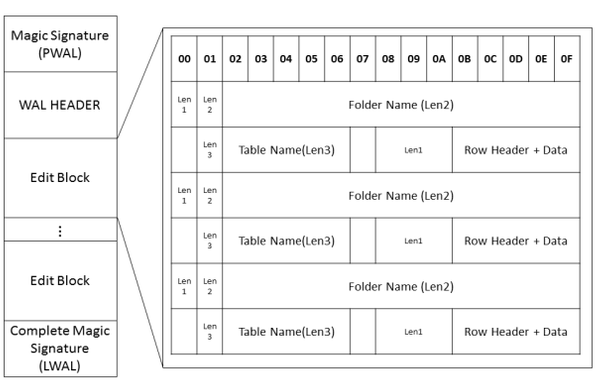

Fig. 8. Structure of WAL

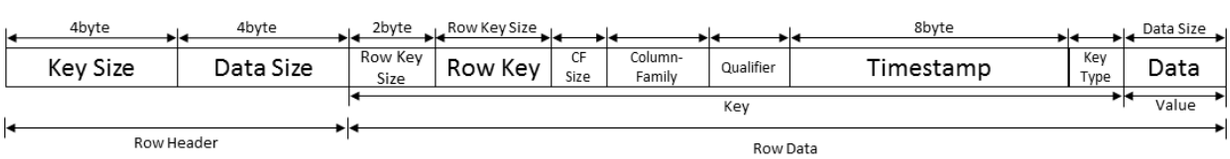

Fig. 7. Structure of HFile (2) - Row Header + Row Data (Column)

5.3 로그파일 수집 및 분석

시스템 오류, 테이블 생성, 데이터 수정 등 시스템이나 데이터베이스 내부의 변경사항이 모두 로그에 기록되므로 사용자의 행위를 유추하기 위해 대상 서버의 로그 파일 수집이 필요하다. 로그 내용은 시간, 로깅 레벨, 행위 등이 있다. 로깅 레벨에는 모든 내용을 기록하는 DEBUG, 데이터 처리를 기록하는 INFO, 오류를 기록하는 WARN과 ERROR, 아무것도 기록하지 않는 None 레벨이 있으며, 데이터의 input/output과 관련된 기록은 대부분 INFO 레벨에 저장되므로 사용자의 행위는 INFO 레벨의 로그를 확인하면 된다. Table 7은 실험 환경에서 수집한 로그 파일로 'posts' 테이블이 생성된 시간이나 데이터를 추가한 시간을 확인할 수 있다.

Table 7. Part of HBase Log

Action	Log
Create Table	2016-07-25 22:36:14,963 INFO [RegionOpenAndInitThread-posts-1] regionserver.HRegion: creating HRegion posts HTD == 'posts', {NAME => 'message', BLOOMFILTER=> 'ROW', VERSIONS=> '1', IN_MEMORY=> 'false', KEEP_DELETED_CELLS=> 'FALSE', DATA_BLOCK _ENCODING=> 'NONE', TTL=> 'FOREVER', COM PRESSION=> 'NONE', MIN_ VERSIONS=> '0', BLOCKCACHE=> 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE=> '0'} RootDir =hdfs:// master:9000/hbase-root/.tmp Tablename==posts
Put Data	2016-08-04 22:21:03,601 INFO [main] http.HttpServer: Added global filter 'safety' (class=org.apache.hadoop.hbase.http.HttpServer$QuotingInputFilter)
Delete Data	2016-08-04 23:54:52,412 INFO [main-EventThread] zookeeper.RegionServerTracker: RegionServer ephemeral node deleted, processing expiration [slave2,60020,1470322236753]

5.4 삭제된 데이터 복구

HBase에서는 테이블에 데이터 추가/수정/삭제 등의 이벤트 발생 시 HFile을 새로 생성하여 변경된 내용만을 기록한다. 이후 일정 시간이 지나면 생성된 HFile들을 하나의 파일로 병합하여 관리한다. 하나의 컬럼은 RowKey, Column-Family, Column-Qualifier로 이루어진 컬럼 Key 값을 가지며, 데이터베이스에 데이터 추가/수정 시 컬럼의 Key 값과 함께 데이터(Value)가 HFile에 기록된다. 하지만 삭제된 데이터가 최신 HFile에 기록될 때는 컬럼 Key 값만이 기록된다.

Fig. 9는 실험 환경에서 추출한 HFile의 구조이다. 각각의 컬럼 Key에는 하나의 KeyType이 존재하고, 이를 이용해 테이블 내에서 삭제된 데이터를 복구할 수 있다.

KeyType이 '0x04'인 경우 테이블 내에 데이터가 추가/수정된 것으로 실제 존재하는 데이터를 의미하고, KeyType이 '0x0C'인 경우 삭제된 데이터를 의미한다. 따라서 원본 HFile과 추가로 생성된 HFile들의 비교를 통해 삭제된 데이터의 유무를 확인한 후 컬럼 Key를 이용해 삭제된 데이터를 복구한다.

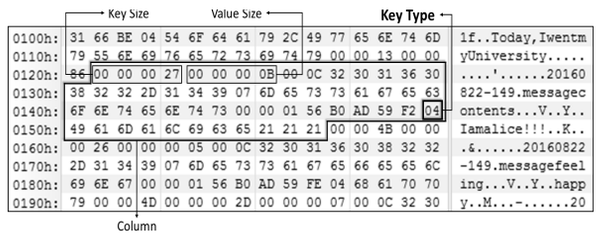

Fig. 9. Detail Structure of HFile

HBase의 삭제된 데이터 복구 방법은 다음과 같다.

먼저, 컬럼패밀리 폴더에 존재하는 HFile의 파일 생성시간이 가장 빠른 파일을 기준으로 나머지 HFile들의 변화를 통해 삭제된 데이터와 정상 데이터를 구분한다. 이후 삭제된 데이터의 컬럼 Key 값을 이용해 데이터(Value)를 찾아 복구할 수 있다.

예를 들어 A라는 데이터가 삭제되었다면, 원본 HFile에 존재하는 컬럼 Key와 Value를 이용해 데이터 A를 복구할 수 있고, 데이터 A가 존재하는 원본 HFile에는 A의 생성시간이 저장되고, 추가로 생성된 HFile에는 데이터 A가 삭제된 시간이 저장되기 때문에 원본 A가 생성된 시간과 A가 삭제된 시간을 알 수 있다.

6. HBase에 대한 디지털 포렌식 조사 기법 검증

Fig. 5에서 제안한 HBase에 대한 디지털 포렌식 조사 기법을 검증하고, 평가하기 위해 실제와 유사한 환경을 구축하고, 실험 시나리오를 만들었다.

6.1 실험 개요 및 시나리오

실험 시나리오는 유언비어 유포에 대한 데이터베이스 포렌식 수사이다. 유언비어 유포자 Trudy는 HBase를 사용하는 SNS에 Alice가 운영하는 빵집에서 탄 빵과 맛없는 빵들을 판다는 유언비어 글을 게시하였고, 문제가 될 것을 염려하여 해당 게시글을 삭제했다. 이 후 Alice의 빵집을 찾는 손님 수는 급격하게 줄어들었고, 뒤늦게 이 사실을 알게 된 Alice에 의해 조사가 시작되었다. 이 시나리오를 바탕으로 본 논문에서 제시한 알고리즘을 이용해 조사관이 다음과 같은 세 가지 질문에 답을 찾아가는 과정을 서술하고자 한다.

- Trudy가 언제 글을 게시했는가?
- 유언비어의 내용은 무엇인가?
- 게시글은 언제 삭제되었는가?

실험은 총 3개의 서버를 이용해 Fully-distributed 모드 환경을 구성하여 진행하였다. 환경 구축을 위해 Hadoop 2.7.2, Zookeeper 3.4.8, HBase 1.0.3이 사용되었으며, 각각의 서버는 SSH 통신을 할 수 있도록 구성했다. 환경 구축과 관련된 자세한 정보는 Table 8과 같다.

실험을 위해 Trudy, Alice, Bob, Charlie 등의 계정을 이용해 글을 작성하여 게시글(posts) Data Set을 구축했다.

Table. 8. The Experimental Environment

No	/etc/hostname	IP Address	SSH	JDK	Hadoop			Zookeeper	Hbase
					NameNode	Secondary NameNode	DataNode		
1	master	192.168.56.140	O	O	O			O	O (HMater)
2	slave1	192.168.56.141	O	O		O		O	O (HRegionServer)
3	slave2	192.168.56.142	O	O			O	O	O (HRegionServer)

6.2 실험 내용

1) 운영환경 파악

6.1에서 제시한 시나리오의 환경을 파악한 결과 다음과 같다. Fig. 10은 HBase Shell에서 명령어 'status'와 'version'을 실행한 결과로 2개의 Region Server가 동작하는 중이고, HBase 버전 1.0.3이 사용되는 것을 알 수 있다.

```
hbase(main):001:0> status
2 servers, 0 dead, 5.0000 average load

hbase(main):002:0> version
1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ea1dd559890,
Tue Jan 19 19:26:53 PST 2016
```

Fig. 10. Information of the HBase on Service

Fig. 11은 HBase Shell에서의 'jps' 명령어의 결과로 'NameNode', 'NodeManager', 'HMaster'의 데몬이 동작하고 있으므로 해당 노드는 HDFS가 동작하는 HMaster 서버임을 알 수 있다.

```
root@master:/home/dbuser# jps
5585 ResourceManager
5618 QuorumPeerMain
5172 NameNode
5750 NodeManager
56726 Jps
55862 HMaster
56125 Main
```

Fig. 11. The Result of Jps Command

Fig. 12는 실험 환경에서 수집한 regionservers 파일과 hosts 파일로 Region Server들은 slave1과 slave2로 분산되었고, 각각의 IP는 192.168. 56.141과 192.168.56.142임을 알 수 있다.

regionservers

```
slave1
slave2
```

hosts

```
192.168.56.140 master  zookeeper0    NameNode
192.168.56.141 slave1  zookeeper1    SecondaryNode
192.168.56.142 slave2  zookeeper2    DataNode01
```

Fig. 12. Regionservers and Hosts Files Extracted from the Case Study

Fig. 13은 실험 환경에서 수집한 'hbase-site.xml' 파일의 일부로 'hbase.rootdir' 속성 값이 'hdfs://'이므로 HDFS를 사용하고, 'distributed' 속성 값이 'true'이므로 분산된 것임을 알 수 있다. 또한 Fig. 12에서 Region Server들의 IP가 다르므로 Fully-distributed 모드임을 알 수 있다.

```
<configuration>
   <property>
       <name>hbase.rootdir</name>
       <value>hdfs://master:9000/hbase-root/</value>
   </property>
   <property>
       <name>hbase.master</name>
       <value>master:60000</value>
   </property>
   <property>
       <name>hbase.zookeeper.quorum</name>
       <value>master, slave1, slave2</value>
   </property>
   <property>
       <name>hbase.cluster.distributed</name>
       <value>true</value>
   </property>
       ...
</configuration>
```

Fig. 13. Part of 'hbase-site.xml'

2) 데이터 수집 및 분석

Fig. 14는 Filter 중 'QualifierFilter'와 'ValueFilter'를 이용해 'posts' 테이블에서 게시글 작성자가 Trudy인 로우를 검색한 결과이며, 총 6개의 로우가 발견되었다. 필터링 결과

```
hbase(main):002:0> scan 'posts', {FILTER=>"QualifierFilter(=, 'substring:name') AND ValueFilter(=, 'substring:trudy')"}
ROW                              COLUMN+CELL
 20160821-189                    column=account:name, timestamp=1471753659459, value=trudy
 20160821-226                    column=account:name, timestamp=1471789161953, value=trudy
 20160822-329                    column=account:name, timestamp=1471863462264, value=trudy
 20160823-453                    column=account:name, timestamp=1471933451632, value=trudy
 20160823-480                    column=account:name, timestamp=1471936604897, value=trudy
 20160824-573                    column=account:name, timestamp=1472049439138, value=trudy
6 row(s) in 0.3660 seconds
```

Fig. 14. The Result of Filtering 'posts' Table

Trudy가 게시한 글은 총 6개가 존재한다.

6.1에서 제시한 시나리오와 같이 Trudy는 본인이 게시한 글을 삭제했으므로 필터링 결과에서 해당 글을 확인할 수 없다. 따라서 삭제된 메시지의 복구가 필요하다.

3) 삭제된 데이터 복구

게시글이 저장되는 컬럼패밀리 명은 'message'이므로 해당 폴더에 존재하는 HFile들을 복구해야 한다. 6.1에서 제시한 문제에서 Trudy가 삭제한 메시지를 복구하기 위해 HFile들의 Key Type을 확인한다. Fig. 15는 실험 환경에서 추출한 원본 HFile이고, Fig. 16은 추가된 HFile의 일부이다. Fig. 16을 확인한 결과 컬럼 Key가 '20160824-573 messagecontents'인 데이터가 삭제된 것을 알 수 있다. 따라서 원본 HFile에서 해당 데이터의 컬럼 Key를 이용해 삭제된 데이터를 복구할 수 있다.

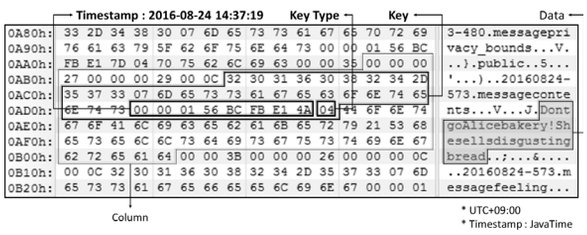

Fig. 15. The Result of Original HFile in Case Study

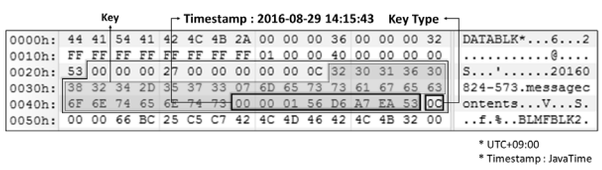

Fig. 16. Deleted Data in Case Study

6.3 실험 결과

본 논문에서 제안한 HBase에 대한 디지털 포렌식 조사 기법을 이용하여 디지털 포렌식 조사관은 다음과 같은 결과를 얻을 수 있다.

Trudy가 게시한 글은 삭제되었으므로 5.4에서 제시한 방법으로 복구한 결과 Fig. 15에서와 같이 원본 HFile에서 '2016-08-24 14:37:19'에 게시된 것을 알 수 있다.

유언비어의 내용은 메시지 복구를 통해 Fig. 15에서와 같이 원본 HFile을 확인한 결과 'Don't go Alice bakery! She sells disgusting bread'라는 것을 알 수 있다.

Trudy가 게시글을 삭제한 날짜와 시간은 Fig. 16에서와 같이 추가로 생성된 HFile을 확인한 결과 '2016-08-29 14:15:43'에 삭제된 것을 알 수 있다.

이처럼 HBase 분석 시 'hbase-site.xml', 'regionservers' 등 설정 파일들을 통해 대상 서버의 운영환경을 파악한 후 필터링 기능을 통해 쟁점이 되는 데이터가 존재하는 HFile의 위치를 파악하여 추출하고, WAL 파일도 해당 폴더에서 파일을 추출하며, 로그파일도 추출한다. 이후 추출된 HFile들을 통해 삭제된 데이터를 복구하여 사건을 분석할 수 있다.

7. 결 론

관계형 데이터베이스(RDBMS)에 대한 디지털 포렌식 조사 기법에 대한 연구는 꾸준히 진행되어 왔다. 기존 RDBMS에서 처리하기 어려운 데이터들이 증가하면서 비관계형 데이터베이스(NoSQL DBMS)의 필요성이 대두되었고, 이에 따라 NoSQL DBMS의 시장 점유율이 증가하고 있으나, NoSQL DBMS에 대한 디지털 포렌식 연구는 미비한 실정이다. NoSQL DBMS에 대한 연구는 문서형 데이터베이스에 대한 디지털 포렌식 연구는 진행되어 왔으나, 데이터베이스의 유형에 따라 데이터의 저장·관리 방법이 다르므로 컬럼형 데이터베이스와 같은 타 제품군에 기존 연구의 절차를 곧바로 적용할 수 없다.

이에, 본 논문에서는 컬럼형 데이터베이스인 HBase에 대한 디지털 포렌식 조사 기법에 대해 연구하였다. 컬럼형 데이터베이스의 특징과 HBase의 구성 및 운영환경과 데이터 모델을 연구하고, 이를 바탕으로 디지털 포렌식 조사 기법을 제안하였다. 또한 데이터 저장 방법에 대한 연구를 통해 삭제된 데이터 복구 방안에 대해 제안하였으며, 실험 시나리오를 통해 제안한 조사 기법을 검증하였다.

향후 HBase 분석의 자동화를 위한 디지털 포렌식 조사 도구를 개발할 계획이며, NoSQL DBMS의 공통적인 요소들을 이용해 NoSQL DBMS의 디지털 포렌식 조사 기법에 대한 연구를 진행할 예정이다.

References

[1] R. Cattell, "Scalable SQL and NoSQL data stores," *Acm Sigmod Record*, Vol.39, No.4, pp.12-27, 2011.

[2] B. Choi, J. H. Kong, S. S. Hong, and M. M. Han, "The Method of Analyzing Firewall Log Data using MapReduce based on NoSQL," *Journal of the Korea Institute of Information Security and Cryptology*, Vol.23, No.4, pp.667-677, 2013.

[3] J. S. Lee and S. C. Hong, "Study on the Application Methods of Big Data at a Corporation," *Journal of the Korea Institute of Information Security and Cryptology*, Vol.15, No.1, pp. 103-112, 2014.

[4] H. K. Khanuja, and D. S. Adane, "A Framework For Database Forensic Analysis," *Computer Science & Engineering: An International Journal (CSEIJ)*, Vol.2, No.3, 2012.

[5] A. Aldhaqm, S. A. Razak, S. H. Othman, A. Ali, and A. Ngadi, "Conceptual Investigation Process Model for Managing Database Forensic Investigation Knowledge," *Sciences, Engineering and Technology*, Vol.12, No.4, pp.386-394, 2016.

[6] K. E. Pavlou, and R. T. Snodgrass, "Forensic Analysis of Database Tampering," *ACM Transactions on Database Systems (TODS)*, Vol.33, Iss.4, pp.1-45, 2008.

[7] J. H. Choi, D. W. Jeong, and S. J. Lee, "The method of recovery for deleted record in Oracle Database," *Journal of The Korea Institute of Information Security & Cryptology(JKIISC)*, Vol.23, No.5, pp.947-955, 2013.

[8] O. M. Adedayo and M. S. Olivier, "Ideal log setting for database forensics reconstruction," *Digital Investigation*, Vol.12, pp.27-40, 2015.

[9] J. S. Yoon, D. W. Jung, C. H. Kang, and S. J. Lee, "Digital Forensic Investigation of MongoDB," *Journal of the Korea Institute of Information Security and Cryptology*, Vol.24, No.1, pp.123-134, 2014.

[10] J. M. Choi, D. W. Jung, J. S. Yoon, and S. J. Lee, "Digital Forensics Investigation of Redis Database," *KIPS Transactions on Computer and Communication Systems*, Vol.5, No.5, pp.117-126, 2016.

[11] F. C., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *Transactions on Computer Systems (TOCS)*, Vol.26 Iss.2, pp.205-218, 2008.

[12] G. Xiaoming, and Q. Judy, "Scalable inverted indexing on NoSQL table storage," Technical Report, Jan., 2013.

[13] S. W. Seo, Hadoop & NoSQL for analyzing and processing big data, in Gilbut, p.401.

[14] T. Harter, D. Borthakur, S. Dong, A. Aiyer, L. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Analysis of hdfs under hbase: A facebook messages case study," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, pp.199-212, 2014.

[15] Hadoop Commands [Internet], https://hadoop.apache.org/docs/r2.4.1/ hadoop-project-dist/hadoop-common/FileSystemShell.html.

[16] D. C. Lee, and S. J. Lee, "Research of organized data extraction method for digital investigation in relational database system," *Journal of the Korea Institute of Information Security and Cryptology*, Vol.22, No.3, pp. 565-573 (9 pages), 2012.

[17] A. B. M. Moniruzzaman and S. A. Hossain, "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison," *International Journal of Database Theory and Application*, Vol.6, No.4. pp.1-13, 2013.

[18] J. S. Yoon, D. W. Jung, C. H. Kang, and S. J. Lee, "Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study," *Digital Investigation*, Vol.17, pp.53-65, 2016.

[19] G. Harrison, "Data Models and Storage," Next Generation Databases, pp.145-166, 2015.

[20] L. George, "HBase : The Definitive Guide, O'Reilly Media, Inc.," pp.333-334, 2011.

[21] H. Zhuang, K. Lu, C. Li, M. Sun, H. Chen, and X. Zhou, "Design of A More Scalable Database System," *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp.1213-1216, 2015.

박 아 란

e-mail : aran725@korea.ac.kr
2015년 숙명여자대학교 멀티미디어과학과 (학사)
2015년~현 재 고려대학교 정보보호대학원 정보보호학과 석사과정
관심분야 : Digital Forensic, Database Forensic, Information Security

정 두 원

e-mail : dwjung77@gmail.com
2011년 고려대학교 산업경영공학과(학사)
2011년~현 재 고려대학교 정보보호대학원 정보보호학과 석·박사통합과정
관심분야 : Digital Forensic, Information Security, Big Data Analysis

이 상 진

e-mail : sangjin@korea.ac.kr
1987년 고려대학교 수학과(학사)
1989년 고려대학교 수학과(석사)
1994년 고려대학교 수학과(박사)
1989년~1999년 ETRI 선임연구원
1999년~현 재 고려대학교 정보보호대학원 교수
2008년~현 재 고려대학교 디지털포렌식연구센터 센터장
관심분야 : Digital Forensic, Steganography, Hash Function