Haddle: a framework for investigating data leakage attacks in Hadoop

Yun Gao[1], Xiao Fu[1]*, Bin Luo[1], Xiaojiang Du[2], Mohsen Guizani[3]
[1]Software Institute, Nanjing University, China
[2]Department of Computer and Information Sciences, Temple University, USA
[3]Qatar University, Doha, Qatar
Email: fuxiao@nju.edu.cn, luobin@nju.edu.cn, xjdu@temple.edu, mguizani@ieee.org

***Abstract*—Nowadays Hadoop is popular among businesses and individuals for its low costs, convenience, and fast speed. However, this also makes it the goal of data leakage attacks as sensitive data stored with an HDFS infrastructure grows rapidly. Therefore, it is important to investigate such attacks in Hadoop. Several works have been done on improving the security of Hadoop, but hardly any have been done on data leakage investigation. This paper presents a typical data leakage attack scene in Hadoop and proposes Haddle (Hadoop Data Leakage Explorer), a forensic framework composed of automatic analytical methods and on-demand data collection based on two stages. With the assistance of Haddle, investigators can find the stolen data, find the perpetrator who stole the data, and reconstruct the crime scene. Also, Haddle can help improve the audit mechanism of Hadoop.**

Keywords—Data Leakage, Cloud Computing, Forensics, Hadoop, HDFS

I. INTRODUCTION

With the emergence and development of cloud computing, more and more people have become used to its convenience. However, security risks like privacy leakage and data breaches have also appeared. Hadoop is a typical model of PaaS and it is popular among businesses and individuals for its powerful processing capacity, huge storage capacity, scalability, and relatively low costs, but "With the growing use of Hadoop to tackle big data analytics involving sensitive data, a Hadoop cluster could be a target for data exfiltration, corruption, or modification." [1] Since Hadoop is usually adopted in large clusters, such as Amazon Elastic MapReduce, where lots of users run their jobs and store their data, it is urgent and important to protect the user data in it.

There exist efforts on strengthening the security of Hadoop, like ACL access control, Kerberos mechanisms, and data encryption. Most of these works have been implemented in Secure Hadoop clusters [2]. However, few works are about investigating the crimes in Hadoop. Moreover, the current audit mechanism in Hadoop, which should be an important source of evidence in the investigation, is not enough for forensics. For example, not only is application level access in HDFS(Hadoop Distributed File System) not registered in Hadoop logs, but the physical storage of data in HDFS is not audited either. Users can thus operate on application level freely without being recorded and their operations on physical level cannot be discovered by Hadoop. For example, a typical data leakage attack in Hadoop can be described as following:

UserA stored file "companyA.7z" in HDFS, and this file is larger than 64M, so it was divided into several blocks, these blocks might be saved in different machines. Hadoop super user and root user have "write" permission to all the blocks. Hence if the perpetrator wants to steal the file, he doesn't have to get the permission in HDFS, he just gets the locations of every block with the help of Name Node logs (or through other ways such as the network monitor and so on…), then steals the blocks directly from those physical machines as he has root permission. For example, if there are userB and userC which are not Hadoop super users, but they have root permission of physical machine, the perpetrator might be one of them. Because the perpetrator got the locations of blocks through Name Node logs and copied the blocks from different Data Nodes directly, it is harder for forensic investigators to find him. The only clue is that userA knows which file was stolen and the Fsimage and other audit logs might be attacked or be tampered with by CSP.

There are several challenges while investigating the case above. The first challenge is a large number of cluster nodes. This is a common challenge faced in cloud computing forensics. Some commercial Hadoop clusters contain hundreds of nodes. When the attack happens in our case, it is almost impossible to investigate all the nodes because either it costs too much time or the evidence has been changed during the investigation. However, most attacks involve only a few nodes, so it is essential to find a way to locate these nodes being attacked and concentrate our resources on them. A second obstacle is the credibility of the evidence. This is the challenge faced by almost all forensic investigations. Any evidence obtained from contaminated data lacks credibility in court. In our case, there might be CSP employees who are authorized to modify Hadoop logs and Fsimage colluding with attackers. Hence, how do we know whether the evidence can be trusted or not and how do we find out the truth if the evidence is untrusted? A final challenge is the lack of available and prerequisite evidence. Due to the uniqueness of HDFS, only Hadoop audit logs can be acquired when crimes happen. However, these audit logs are far from sufficient in investigations as they lack the information on "who" did the crimes, furthermore, these audit logs can be easily tampered

This work is supported by the National Natural Science Foundation of China (61100198/F0207, 61100197/F0207).

with by attackers. Hence, other ways should be found to acquire more effective evidence.

By studying Hadoop audit logs and Mark Will's "Progger" [3] project, which can audit all the user operations in Linux, we propose Haddle (Hadoop Data Leakage Explorer), a forensic framework composed of Data Collector and Data Analyzer. Data Collector has a two-stage collection according to the specific scenario. It collects Hadoop logs, Fsimage files [4], Hadoop-Progger [3] logs, and other information from client machines and transfers them to the server. Then Data Analyzer analyses the data with automatic analytical methods to find the stolen data, find the perpetrator who stole this data, and reconstruct the crime scene. The automatic detection algorithm in Data Analyzer detects data leakage attacks from four dimensions including abnormal directory, abnormal user, abnormal operation, and block proportion. With the help of Haddle, the automatic detection algorithm can find abnormal actions quickly and locate the attacks nodes, then notify the Collector Engines of those nodes. In this manner, no matter how big the Hadoop cluster is, investigators only have to concentrate on a few nodes. On the other hand, with the help of Hadoop-Progger, investigators can get plenty of evidence sources and a reliable evidence chain, even if someone tampers with the Hadoop logs or Fsimage files.

II. RELATED WORK

Current Hadoop security research mainly includes trusted architecture creation, access control design, data encryption, and so on. Forensics in Hadoop hasn't received much attention.

In [1], an IT architecture was proposed to fight against Advanced Persistent Threat (APT) targeted on data stored in HDFS. This IT architecture is based on trusted computing concepts and technologies like the Trusted Platform Group (TPM) and Trusted Computing Group (TCG). With the help of this IT architecture, all of the operations triggered by any user can be audited. In this way, suspicious actions can be discovered and evidence can be retrieved for future forensics. Thus, a more robust security posture can be achieved, however, it results in a serious impact on performance and a huge size of logs, which make it unrealistic in practice.

In [2], a complicated access control mechanism was adopted to ensure the security of Hadoop. These access control mechanisms protect data in Hadoop from unauthorized data access, accidental data leakage and loss, and breach of tenant confidentiality to make Hadoop securer. "ACL Access Control" and "Kerberos" are two of them, which are adopted by the new versions of Hadoop. Although this can make Hadoop securer, it can't solve the case presented before. For example, the Hadoop system is not able to identify perpetrators if they use legal user accounts for application level attacks, and for operating system level attacks, nothing can be done as these access control mechanisms are only for the Hadoop application level.

In [5], a secure Hadoop architecture by adding encryption and decryption functions in HDFS was proposed. Through this method, data in HDFS won't be readable even if it is stolen because the perpetrator doesn't have the secret key. Data is protected in this way. Although it is a fundamental solution for securing Hadoop, what can't be ignored is the impact on performance and the helplessness of Hadoop application-level attacks. On the other hand, encryption makes it much more difficult in data mining and analysis.

The research associated with Hadoop forensics mainly concentrates on making use of Hadoop and its derivative products to help promote storage capacity, analysis capacity, and processing capacity [6-9].

Unlike these works, our Haddle concentrates on investigating data leakage attacks in Hadoop. We haven't found any other work on such a problem.

III. HADDLE DESIGN

Haddle is composed of Data Collector and Data Analyzer. In this section, the design of each will be presented.

A. Data Collector Design

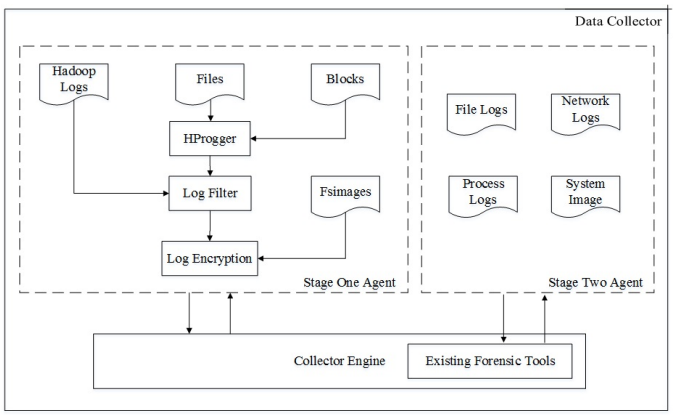

Fig. 1. Architecture of Data Collector.

Data Collector is installed in the node of Hadoop. The architecture of Data Collector is shown in Figure 1. The whole process of data collecting contains two stages: Stage One is the main work that this paper will focus on and Stage Two is the applying of existing forensic tools which can be used directly. In Stage One, the Collector Engine tells the Stage One Agent to collect Hadoop logs, Hadoop-Progger logs, and fsimage files. Then, Log Filter formats the logs to a unified standard. Finally, Log Encryption transmits them to the server after encryption. Stage Two depends on the results of Stage One; existing forensic tools will be employed to collect file logs, network logs, process logs, and system images and transmit to the server for analyzing. The two-stage design gives Haddle the ability to collect data according to what the investigation needs, which causes reduced consumption and improved efficiency. To complete these steps, we need the following modules: Log Filter, Log Encryption, and Collector Engine.

Before presenting the design of each module, the data source of Haddle should be introduced first. The data source of Haddle includes Hadoop logs, Fsimage file, and HProgger logs. Hadoop logs are the audit logs generated by the Hadoop system. These logs record the operations conducted by Hadoop users through Hadoop commands. There are several types of Hadoop logs, including Name Node logs, Job Tracker logs, Data Node logs, and Task Tracker logs. As the attacks in our case happen only in HDFS, so Haddle will mainly focus on Name Node logs and Data Node logs. Fsimage file is the image file of the Hadoop system. Unlike general system image files, fsimage only records the data directory and data distribution. In simple terms, fsimage records the Meta information of files in file system and the distribution of blocks. The block is the basic unit of HDFS files that is stored in physical machines. Each file is composed of one or more blocks according to the size of the file. In our framework, Hadoop logs are collected in real time by modifying the log4j configuration file. In this way, perpetrators will have no chance to tamper or delete them as they are transferred to Log Filter directly. As for Fsimage file, it is uploaded to Data Analyzer periodically.

HProgger is the improved version of Progger [3]. Progger is a forensic tool proposed by Mark Will. When loaded, the module modifies the addresses of certain system calls in the system call table allowing Progger to create logs when these calls are accessed. Hence, whenever there is an operation, Progger can find almost all important information about this operation through Progger logs. As current investigations in Hadoop lack suitable evidence, Progger is able to help us acquire more reliable evidence because any data leakage attack in Hadoop will inevitably involve operations on files and directories. Although Progger is powerful in auditing, it causes a serious downgrade on performance and delays every operation. Additionally, the amount of logs that Progger generates is unacceptable; it generates hundreds of megabytes of logs in just a few minutes. To make it applicable in our Haddle, performance must be improved and the amount of logs must be reduced to the greatest extent. The improved version of Progger is called "HProgger" (Hadoop-Progger). HProgger improves several aspects. (1) Only a few system call types will be recorded. Table I shows these types. (2) Log format is minimized, only necessary information like operation type, time, operator, and so on remains. (3) Instead of monitoring all of the directories, only those associated with the Hadoop system, like the directory where Hadoop stores HDFS files, are retained. Hence, compared to Progger, HProgger is lighter and more applicable. Like Fsimage file, HProgger logs are uploaded to Data Analyzer periodically. This may cause security problem as perpetrators may tamper or delete HProgger files during a period, so real-time mechanism will be given in future work.

TABLE I. OPERATION TYPES LOGGED IN HPROGGER

Operation Type	Description
OPEN	When someone opens or accesses a file, an "HProgger: 0" type log will be recorded. This kind of operation is called OPEN operation in this paper.
COPY	When someone copies a directory or file, an "HProgger: 3" type log will be recorded. This kind of operation is called COPY operation in this paper.
MOVE	When someone moves a file from one directory to another or renames a file, an "HProgger: 4" type log will be recorded. This kind of operation is called MOVE operation in this paper.

Log Filter formats the logs to a unified standard, as it is important to control the sizes of logs and extract useful information. A lot of experiments and statistical analysis on different logs have been done to find out the most important and most useful data. Finally our standard format has been developed according to the following format:

"HProgger, node ip, time, operation type, user name, directory 1, directory 2, file name 1, file name 2."

(1) Node ip is the ip address of node machine;

(2) Operation types include OPEN, COPY and MOVE;

(3) User name logs the system name of operator;

(4) Directory 1 logs the old directory and 2 logs the new directory;

(5) File name 1 logs the name of old file and 2 logs the name of new file.

Log Encryption encrypts the logs before they are transmitted to the server. This helps improve our own security measures and make our evidence more credible.

Collector Engine is the trigger of Data Collector and ties the clients with the server. To solve the challenge of large numbers of cluster nodes making our investigations sluggish and inefficient, Data Engine is designed. It connects the client and the server so that the server can take control of the investigation process. Data transmitted from Data Engine to the server will be marked by an explicit node, hence when the investigator digs out some suspect records, they are able to locate the nodes being attacked according to the marked data quickly and continue Stage Two data collecting by Data Engine in those nodes instead of all nodes.

B. Data Analyzer Design

Data Analyzer is composed of Log Database, Log Manager and Automatic Detection Module. It is installed in the forensic server. Log Database stores the logs. Users can query and manage logs in Log Manager and Automatic Detection Module analyses the logs automatically to find the data that was stolen, the perpetrator who stole the data, and the process of the attacks. The results will be displayed in Log Manager.

In some cases, people may not realize the data leakage immediately. Hence, how can Haddle discover the attack as

soon as possible after it happens? Here, an automatic detection algorithm is introduced.

1) Definitions

File Dataset: A file consists of one or more blocks in HDFS and every block has a unique block ID. HProgger records extracted according to all block IDs that belonging to a file are called a file dataset.

Block Dataset: HProgger records extracted according to a block ID are called a block dataset.

Suspicious Block: If some abnormal information occurred in records that associated with a specific block, then this block is a suspicious block.

2) Automatic Detection Algorithm

Our algorithm consists of four dimensions: abnormal directory, abnormal user, suspicious block proportion, and abnormal operation. The algorithm detects abnormal directories and abnormal users in file datasets to find out if suspicious blocks exist and it calculates the suspicious block proportion. In the meantime, the algorithm monitors the trend of operations in block datasets to detect abnormal values and also calculates the suspicious block proportion. If any of these four dimensions gives a warning, then attacks may have happened and investigators can continue their investigation according to the warning information.

Abnormal Directory (AD). Normal Hadoop system file operations will involve fixed directories because these directories are configured by Hadoop system profiles. Hence, if any directory in a file dataset that is out of this range is found, an attack may happen. The suspicious blocks can thus be found by analyzing records that contain these abnormal directories. By studying the HProgger logs generated by all kinds of Hadoop system operations, all of the fixed directories can be found and shown in Table II.

TABLE II. FIXED DIRECTORIES IN HADOOP SYSTEM.

Directory	**Description**
/app/hadoop/tmp/dfs/data/current	Directory where Hadoop system stores all the HDFS files.
/app/hadoop/tmp/dfs/data/blocksBeingWritten	Directory where Hadoop system stores temporary files when it writes files into HDFS or runs map reduce jobs.
/usr/local/hadoop	Root directory of Hadoop system.

Let AD represent the total number of directories involved in a file dataset. Table II shows that the normal value of AD is 3, hence if the algorithm detects the value of AD greater than 3, attacks may have happened.

$$AD = DirectoryNumber\ (FileID)$$

Abnormal User (AU). By studying the HProgger records, it can be concluded that all of the Hadoop system file operations except copying files from HDFS to local machine will involve only one operator, the Hadoop super user. Hence, if any other user instead of the Hadoop super user is found in a file dataset, an attack may happen. The suspicious blocks can thus be found by analyzing records that contain these abnormal directories. On the other hand, if a perpetrator wants to steal blocks directly from the operating system, he must have root permission or Hadoop super user permission. However, because more than one user can switch to root, how can the real user who conducted operations in root permission be found? HProgger is able to keep track of all users' data as long as they login to the system [3], so the HProgger logs will record the real user instead of root.

Let AU represent the total number of users involved in a file dataset. As Hadoop super user is the only legal user, the normal value of AU is 1, hence if the algorithm detects a value of AU greater than 1, attacks may have happened.

$$AU = UserNumber\ (FileID)$$

Abnormal Operation (AO). Every Hadoop system operation in HDFS is composed of reading files from HDFS and writing files into HDFS. By studying the HProgger logs generated by all kinds of Hadoop system operations, it can be concluded that every block is written only once during its life cycle. Table III shows the mappings between Hadoop operations and HProgger log types.

TABLE III. MAPPINGS BETWEEN HADOOP OPERATIONS AND HPROGGER LOG TYPES

Hadoop Operation	**HProgger Log Type**
write file into HDFS	1 OPEN & 1 COPY & 1 MOVE
read file from HDFS	1 OPEN

If an attacker wants to steal a block from the physical machine, he has to copy (move) it to another directory and sometimes may rename the block. Table IV shows the mappings between operating-system-level operations and HProgger log types.

TABLE IV. MAPPINGS BETWEEN OPERATING SYSTEM LEVEL AND HPROGGER LOG TYPES

System Operation	**HProgger Log Type**
copy file	1 OPEN & 1 COPY
move file	1 OPEN & 1 MOVE
rename file	1 OPEN & 1 MOVE
Read file	1 OPEN

Let AOm represent the number of MOVE in a block dataset.

$$AOm = MoveNumber\ (BlockID)$$

Let AOc represent the number of COPY in a block dataset.

$$AOc = CopyNumber\ (BlockID)$$

Assume the replication number of block in Hadoop system is R, the normal value of both COPY and MOVE in a block dataset is R. If either AOm or AOc is greater than R, someone has stolen (copied or moved or renamed) this block.

This dimension detects all the HProgger logs in every block dataset.

Block Proportion (BP). Let SusBlockNum (FileID) represent the number of suspicious blocks in a file dataset and AllBlockNumber (FileID) represents the total number of blocks in a file dataset.

$$BP = SusBlockNum(FileID) / AllBlockNumber(FileID)$$

Under normal circumstances, the value of BP should be 0 as no suspicious block should be found. The larger BP is, the greater the probability of attack is, so if BP is larger than 0, the algorithm will give a warning.

Algorithm 1 shows the core of our automatic detection algorithm.

ALGORITHM 1. AUTOMATIC DETECTION ALGORITHM

```
1:  function AttackDetection(FileID)
2:      S ← 0
3:          AD ← DirectoryNumber (FileID)
4:          if AD > 3  then
5:                      S++
6:              end if
7:              AU ← UserNumber (FilrID)
8:          if AU > 1  then
9:                      S++
10:             end if
11:     for i = 0 → BlockNumber (FileID)  do
12:             AOm ← MoveNumber (BlockID)
13:             AOc  ← CopyNumber (BlockID)
14:             if Om > R or Oc > R then
15:                     S++
16:             end if
17:     end for
18:     BP ← SusBlockNum(FileID) / AllBlockNumber(FileID)
19:     if BP > 0
20:              S++
21:     end if
22:     if S == 0  then
23:             return normal
24:     else
25:             return warning
26:     end if
27: end function
```

S: number of dimensions whose value exceeds the normal value; AD: value of dimension abnormal directory; AU: value of dimension abnormal user; AOm: number of MOVE in dimension abnormal operation; AOc: number of COPY in dimension abnormal operation; R: replication number of block in hadoop; BP: value of dimension block proportion.

Step1. The algorithm starts by detecting abnormal directories. It calculates the number of total directories in a file dataset. If it is greater than 3, then this dimension is abnormal.

Step2. The algorithm calculates the number of total users in a file dataset. If it is greater than 1, then this dimension is abnormal.

Step3. The algorithm calculates the number of total MOVE and COPY operation in every block dataset belonging to the file dataset. If either number of MOVE or number of COPY is greater than R, this dimension is abnormal.

Step4. The algorithm calculates the proportion of suspicious blocks in a file dataset. If it is greater than 0%, this dimension is abnormal.

Step5. If the number of abnormal dimensions is larger than 0, the algorithm gives the user warnings and investigators can continue their investigation according the results of the warning information.

IV. EXPERIMENT

A. Experiment Preparation

Cluster Environment: A small Hadoop cluster using Virtual Box including 1 master node and 15 slave nodes is set up. (Haddle can work on clusters composed of hundreds of nodes, however, due to the limitation of our experiment environment, experiments on more nodes couldn't be implemented. This will be achieved in our future work.)

Node Environment: i5-4590 3.3GHz 2G Ubuntu 12.04

Hadoop Version: 1.2.1

B. Experiment Results

The scene described in our data leakage attack case is reconstructed. Furthermore, this time userA doesn't know that his data has been stolen. Haddle tries to find out the attack automatically using our algorithm.

The automatic detection algorithm runs every 5 minutes in our experiments (the frequency can be customized). Fig.3 shows the monitoring results of "companyA.7z" during a period of half an hour.

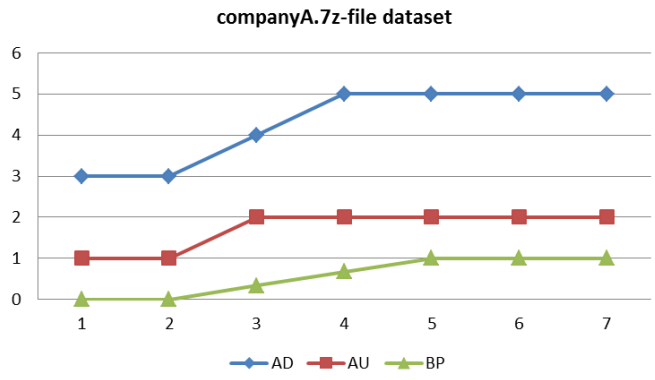

Fig. 2. Experiment results of "companyA.7z"— file dataset.

Fig.2 shows that all three dimensions of file "companyA.7z" changed from timestamp 2 to 5 and all values of the three dimensions are greater than their normal value. For other files, the values of all dimensions remained unchanged the whole time.

Fig.3 shows the values of all three blocks of "companyA.7z" growing to 4 or 5, both greater than the normal value 3. For blocks of other files, the values remained at 3 all the time.

Hence, suspicious operations must have happened to "companyA.7z" during timestamp 2 to 5.

So the detailed suspicious HProgger logs were found, which showed that userB copied block -6299596445748830000 and block -713202901267065000 from "/app/hadoop/tmp/dfs/ data/current" to "/home/abc/" and she copied block 40679224 87609870000 to "home/userB" and then to "home/abc". All the blocks of "companyA.7z" have been stolen by userB. Hence, Haddle finally found the perpetrator, found the stolen file, and reconstructed the crime scene.

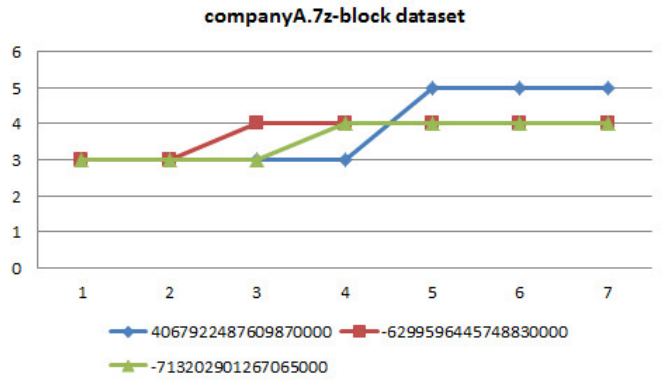

Fig. 3. Experiment results of "companyA.7z"— block dataset.

V. Conclusion and Future Work

This paper presents a forensic framework including automatic analytical methods and on-demand data collection based on two stages. Haddle collects this data from every machine in the Hadoop cluster to our forensic server and analyzes them. With an automatic detection algorithm, Haddle calculates values of four dimensions and analyzes them to find out whether there exist suspicious behaviors and gives warnings and evidence to users if there are. While, Haddle is now designed for data leakage attacks in Hadoop, more types will be covered in the future.

However, there still exist some limitations that should be improved. In first stage forensics, a better way needs to be discovered to collect logs and files from all of the machines in the Hadoop cluster to make data more reliable and reduce the impact on performance. Moreover, our attack detection algorithm has to be improved by studying more real scenes and analyzing more data to find out a more accurate and more efficient way.

References

[1] Jason Cohen, Subrata Acharya. "Towards a More Secure Apache Hadoop HDFS Infrastructure," in Network and System Security, Lecture Notes in Computer Science Volume 7873, 2013, pp 735-741.

[2] Devaraj Das, Owen O'Malley, "Adding Security to Apache Hadoop", Hortonworks Technical Report, 2011. [Online]. Available: http://br.hortonworks.com/wp-content/uploads/2011/10/security-design_withCover-1.pdf

[3] Ryan K. L. Ko, Mark A. Will. Progger: An Efficient, Tamper-Evident Kernel-Space Logger for Cloud Data Provenance Tracking. Presented at 7th International Conference on Cloud Computing (CLOUD), 2011 IEEE. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6973827

[4] D.Borthakur. The Hadoop Distributed File System: Architecture and Design. The Apache Software Foundation, 2007.

[5] Seonyoung Park and Youngseok Lee, "Secure Hadoop with Encrypted HDFS," in GPC, Seoul, Korea. May 9-11, 2013. Proceedings.

[6] Jooyoung Lee, Do won Hong. Pervasive Forensic Analysis based on Mobile Cloud Computing. Presented at Third International Conference on Multimedia Information Networking and Security 2011. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6103838

[7] Taerim Lee, Hun Kim. Implementation and Performance of Distributed Text Processing System Using Hadoop for e-Discovery Cloud Service. Journal of Internet Services and Information Security 2013. [Online]. Available: http://isyou.info/jisis/vol4/no1/jisis-2014-vol4-no1-02.pdf

[8] Vassil Roussev, Liqiang Wang. A cloud computing platform for large-scale forensic computing. 7th IFIP WG 11.9 International Conference on Digital Forensics 2009. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-04155-6_15

[9] Jakrarin Therdphapiyanak, Krerk Piromsopa, "Applying Hadoop for log analysis toward distributed IDS," in ICUIMC, New York, NY, USA. 2013. Proceedings.