*Journal of Computer Science & Computational Mathematics, Volume 7, Issue 3, September 2017*
DOI: 10.20967/jcscm.2017.03.007

# Enhancement Data Integrity Checking Using Combination MD5 and SHA1 Algorithm in Hadoop Architecture

Yaakub Bin Idris*, Saiful Adli Ismail ,Nurulhuda Firdaus Mohd Azmi, Azri Azmi, Azizul Azizan

Avanced Informatics School, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra,
54100 Kuala Lumpur, Malaysia
*Corresponding author email: yaakub.idris@gmail.com

***Abstract***: The use of Big Data in decision-making is critical, in line with the growing size of data storage, either online or offline. However, there are only a few software applications that are capable to process large-capacity data such as Hadoop. Hadoop is open-source software for Big Data processing including several components joined together where one of its main components is Hadoop User Experience (Hue). Hue is being used to upload the data into Hadoop databases using Graphical User Interface (GUI). However, Hue is not equipped with a function to evaluate whether the downloaded data has changed or not, resulting in the processing of incorrect data that leads to false decisions. Therefore, this study aims to improve the functions available in Hue using MD5 and SHA1 cryptographic functions for data verification purposes. These cryptographic functions have been chosen due to their acceptance worldwide and with the added functionality of data verification in Hue, data validation can be performed during uploading process to prevent users from processing erroneous data. The result of this study will ensure the integrity of the data by validation in any means of changes of data before being stored to the Hadoop in offline mode.

***Keywords***: Big Data, data integrity, Hadoop architecture, Hadoop user experience, cryptography**.**

## 1. Introduction

In digital eras, data capacity keeps on increasing without any confines due to rapid improvement of technology. Big Data is the term for data sets that are so enormous and complex, where common data processing applications are inadequate to deal with these data sets [1]. Data driven decisions are critically important in many application domains. Data has become a highly valued resource and requires appropriate security assurances [2]. A major concern in data security is the assessment of data integrity in untrusted servers with respect to large data in cloud platforms [3].

Due to large sets of data transferred from one node to the other, data detection is needed to check if there have been modifications to the data being imperative to ensure data integrity. There are two ways in which data might be altered: firstly accidentally, through hardware and transmission errors, and secondly because of a cyber-attack [4]. The aim of the data integrity service is to detect whether data has been altered or not, however it does not consider data restoration to its original state.

Hadoop user experience (Hue) is one of the modules that allows users to upload data to Hadoop Distributed File System (HDFS) without logging into a Hadoop gateway hosted by using a terminal program based on command line prompt [5]. Due to Big Data issues that need to be considered [6], data integrity is becoming more critical. Furthermore it is easy to modify data when the transfer is being done from one node to the other. The challenges of the Big Data ecosystem are categorized into four main aspects, as illustrated in Figure 1 namely Infrastructure Security, Data Privacy, Data Management and Integrity and Reactive Security.

## 2. Literature Review

Data security is divided into three main dimensions namely confidentiality, integrity and availability. This paper focuses on data integrity where the data is required to be intact and maintained intact. Nowadays, integrity of data storage can be verified by traditional systems through deploying digital signatures, checksums, message authentication code, Reed–Solomon code, trapdoor hash functions, etc. Unfortunately, when the data size increases due to the continuous advancement of the device storage technology, such methods are inefficient from the perspective of the time consumed to process and the required communication speed [7].

This paper specifically focuses on checking the validity of the data instead of recovering back the corrupted or violated data during uploading into the HDFS. In ensuring data integrity, there are three common data integrity-checking methods that exist and are being currently implemented [8]. The techniques used to verify the integrity are achieved by recalculating or re-computing the copied data from the original data and matching it with the information that is being copied. They include Mirroring, Redundant Array of Independent Disks (RAID) Parity [9] and Checksums [10, 11]. Based on the three most common integrity assurance techniques mentioned, the checksum technique is chosen for this study due to its ability to compare the data with the actual data in term of integrity checking.

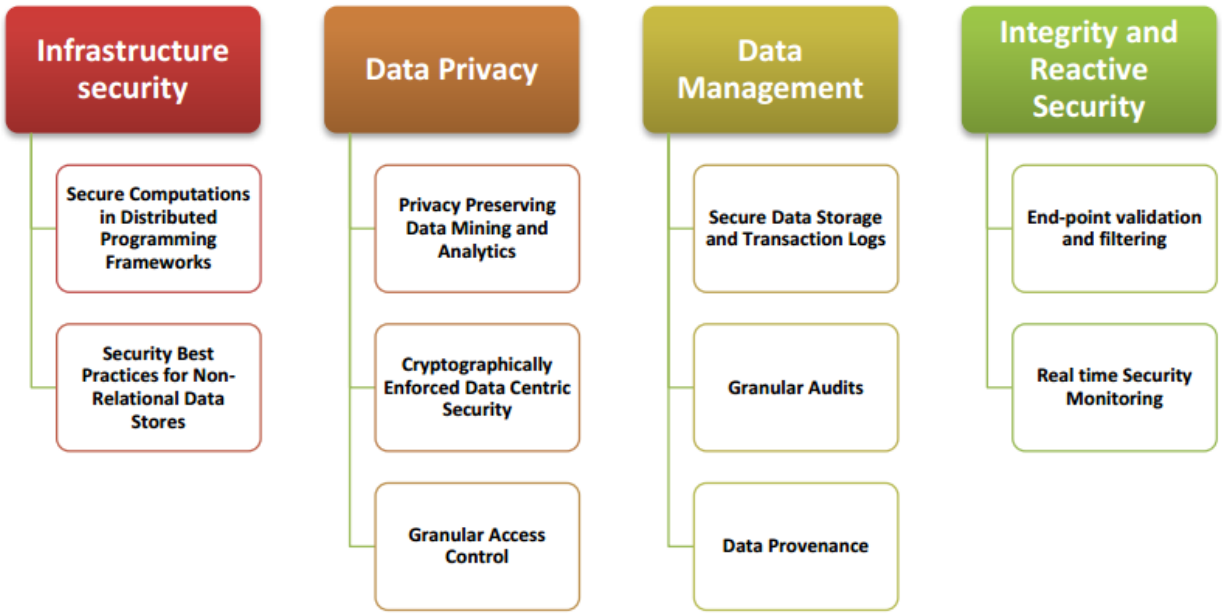


**Figure 1.** Classification of the Challenges of the Big Data Ecosystem

## 3. Cryptographic Hash Functions

To date, cryptographic hash functions have been accepted globally as a standard that is commonly used in protocols and Internet applications [10]. These functions are generally designed to be collision resilient, which means that finding two strings that have the same hash result should be quite impossible [12]. In addition to basic collision resistance, it is noted that two hash functions i.e. the Message Digest 5 (MD5) [13] and Secure Hash Algorithm (SHA1) [14] have noteworthy randomness properties.

Every cryptographic hash function is a hash function, but not every hash function is a cryptographic hash. A cryptographic hash function aims to guarantee a number of security properties and the most importantly that it is hard to find collisions and that the output are based on the algorithm that is being chosen. Typical hash functions or non-cryptographic hash functions are intended to avoid collisions for non-malicious input. Some functions aim to detect accidental changes in data "cyclic redundancy checks" (CRC). Below are the lists of some well-known hash functions [15].

The cryptographic hash functions include:

- SHA series (sha, SHA1, sha-224, sha-256, sha-384, sha512)
- MD series (md2, md4, MD5)
- Tiger
- HAVAL
- RIPEMD series (RIPEMD-128, RIPEMD-160, RIPEMD-320).

The typical hash functions or non-cryptographic hash functions are:

- CRC series (crc16, CRC32, crc64)
- Simming (sum8, sum16, sum24, sum32, xor8).

Table 1 shows the comparison between MD5 and SHA1, to inform the users on which hash functions need to be selected, MD5 or SHA1 or both MD5 and SHA1. To note, both algorithms have different architectures but they follow the same concept [16]. It is observed that, in term of security aspect, SHA1 is more secured than MD5. This is because SHA1 has more message digest length compared with MD5. However if the processing speed for validating the offline data is the main goal, then MD5 will be the best selection. Even though there have been successful attacks reported for MD5 and not for SHA1, that does not mean that SHA1 or any other hashing algorithm is invulnerable to attacks as new upcoming attack techniques are improvised. For this study, even if MD5 or SHA1 has been successfully attack reported, it is still a daunting task for the attacker if MD5 and SHA1 are being used for validation concurrently, compared to validating only MD5 or SHA1. Herein, we propose a given pseudo code that is implemented, either by using MD5 or SHA1 or both MD5 and SHA1 for validating the offline data.

## 4. Current Design Architecture

Hue is one part of Hadoop, which contributes the most important components. It helps users to significantly increase ease of access to the powerful Hadoop platform instead of using Command Line Interpreter (CLI). While YARN and Hive provide a processing backbone for data analysts who are familiar with SQL to use Hadoop, Hue provides the interface for analysis of the data hence rapidly is connected to Hadoop and its big data components architecture. The current existing interface only provides data uploading function directly to the database, without checking the data integrity, therefore the data that is going to be processed might results into erroneous end results, if the data has been tampered as shown in Figure 2. The results are crucial especially if the data are being used as inputs for decision maker.

**Table 1.** Comparison between MD5 and SHA1

| Key For Comparison | MD5 | SHA1 |
|---|---|---|
| **Security** | Less Secured than SHA | More Secured than MD5 |
| **Message Digest Length** | 128 Bits | 160 Bits |
| **Speed** | Faster, only 64 iterations | Slower than MD5, Required 80 iterations |
| **Successful attacks so far** | Attacks reported to some extents | No such attach report yet |
| **Attacks required to find out original Message** | $2^{128}$ bit operations required to break | $2^{160}$ bit operations required to break |
| **Attacks to try and find two messages producing the same MD (Message Digest)** | $2^{64}$ bit operations required to break | $2^{80}$ bit operations required to break |


**Figure 2.** Existing Architecture For Hue

## 5. Proposed Architecture Design

As noted earlier the existing interface of Hue is only to provide uploading function directly to the database without checking for integrity. Hence to make sure that the offline data that will be uploaded into Hadoop has not been altered (during the transaction when the Hadoop user environment is used) or will be altered accidentally, (because of an attack or through hardware and transmission errors), the current existing Hue existing architecture needs to be enhanced.

Figure 3 shows the proposed architecture for an enhanced Hue with all of the additional data integrity components. It shows how the data is going to be validated throughout the purposed module, since it is difficult to detect any changes to the offline data especially during uploading process from the client to the server. Checking for data integrity using MD5 and SHA1 is proposed to be developed in the middle of the current architecture which is between the user interface and the database. There will be a checking module for data integrity to detect any changes to the data where the data is not directly loaded to the database as conventionally implemented.

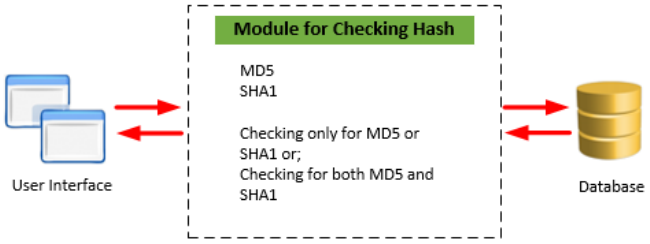


**Figure 3.** Proposed Architecture for the Enhanced Hue

Below is the proposed pseudo code that is going to be implemented.

*Upload Upload the file into the system*
*Enter hashing provide from the original source*
*If a check for integrity using MD5 and SHA1 pass,*
*Notify message pass*
*Else*
*Notify message fails*

## 6. Research Contributions

The main contribution of this study is to prevent data from being tampered especially in Big Data environment, where data is used to analyse, to determine decisions or to predict solutions. Data integrity as a state, describes a set of data that is both accurate and valid, while data integrity as a process, defines the processes used to make sure accuracy and validity of the set of data (or the entire data) contained in a database or construct.

This research contributed the data integrity module using MD5 and SHA1. In addition, it is an implementation through GitHub, hence, this module is coded and can be shared globally where it can be used for all versions of Hadoop. Currently the Hue project is being shared to GitHub, where it has 89 contributors that actively contribute towards enhancing the Hue project where the numbers of contributors are expected to increase.

## 7. Limitation and Challenges

Emphasizing out the existing interface of Hue only by providing uploading function directly to the database without integrity checking, this paper is going to propose another layer for checking before data is being uploaded into Hadoop using MD5 or SHA1 or both MD5 and SHA1 regardless other issues such as effectiveness performance of selected hashing techniques that will be discussed in future work.

Nevertheless, there have been collision attacks towards cryptographic hash functions and increasing numbers of researchers are evaluating the attacks on Hash Functions of MD5 [17] and SHA1. These attacks are targeted to MD5 and SHA1 due to Common Hashing Algorithms being used around the world. The detail of the technical documentation can be accessed through “Hash functions: Theory, attacks, and applications” [18] whereby, the hash has shown

resiliency, that it has remained the same, even though the data has been changed. A review of comparative study of MD5 and SHA security [19] has also been done in order to make the data transmission more reliable and secure by juxtaposing the two hash algorithms of MD5 and SHA, using various key features and performance metrics.

To address this attack, this proposed architecture has a function that can check both MD5 and SHA1 for any changes to the data. This is due to the probability for getting same hashing value [19] when comparing hashing values for MD5 and SHA1 in this research are impossible, rather than comparing individually MD5 or SHA1.

## 8. Conclusion and Future Work

Data integrity is the property whereby data has not been altered in an unauthorized manner, when it was created, transmitted, or stored by an authorized source. Maintaining the integrity of the data or information is one crucial element in Big Data where checksums and cryptographic checksums, are utilized for verification of integrity.

This research is a pilot study for other data validation technique where several other methods can be looked into, either by using Mirroring technique, Checksum technique, Hamming code method, Hash function method and Message Authentication Code Method. These offline methods perform integrity checking in scheduled intervals of time. They are several unkeyed cryptographic hash functions and the numbers will keep increasing due to the needs of securing the data. The types of the hash functions have their own advantages based on the type of data that needs to be protected.

From the security viewpoint, online integrity checkers are better than offline ones. However, online methods come mostly with performance costs. Performing operations like checksum comparison in the critical section of a file system read, could slow down the system noticeably. Offline integrity checkers generally run in an asynchronous manner and hence do not pose such performance problems. For the time being, the proposed enhanced Hue can validate the changes of data before it is stored to the Hadoop in offline mode. As attackers will eventually find a way to bypass all the security measures, other techniques like adding salt, using public key or private key can be used to make the tempering process much harder. For the time being, the scenario assumed in this study is that the attacker can change MD5 or SHA1 value without changing the hashing value. By using the proposed enhanced Hue, the attacker will not be able (or it will be quite difficult) to produce both MD5 and SHA1 (the same value) if both validation checking of MD5 and SHA1 take place.

## References

[1] N. Sheikh, "Big Data, Hadoop, and Cloud Computing", *Implementing Analytics*, pp. 185–197, 2013.

[2] A. Vera-Baquero, R. Colomo-Palacios, O. Molloy, "Towards a Process to Guide Big Data Based Decision Support Systems for Business Processes", *Procedia Technology*, vol. 16, pp. 11–21, 2014.

[3] Institute of Electrical and Electronics Engineers (Ed.), in 2012 *International Conference on Advances in Engineering, Science and Management (ICAESM 2012)*: Nagapattinam, Tamil Nadu, India, 30 - 31 March 2012. Piscataway, NJ: IEEE.

[4] B. B. Madan, Y. Lu, "Attack Tolerant Big Data File System", in *Sigmetrics Big Data Analytics Workshop*, Citeseer, 2013.

[5] B. Saraladevi, N. Pazhaniraja, P. V. Paul, M. S. S. Basha, P. Dhavachelvan, "Big Data and Hadoop-a Study in Security Perspective", *Procedia Computer Science*, vol. 50, pp. 596–601, 2015.

[6] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, S. Ullah Khan, "The rise of 'big data' on cloud computing: Review and open research issues", *Information Systems*, vol. 47, pp. 98–115, 2015.

[7] Q. Xu, G. Liu, "Configuring Clark-Wilson Integrity Model to Enforce Flexible Protection", in Computational Intelligence and Security, 2009. CIS '09, Beijing, China, 2009, pp. 15–20.

[8] G. Sivathanu, C. P. Wright, E. Zadok, "Ensuring data integrity in storage: Techniques and applications", in *Proceedings of the 2005 workshop on Storage security and survivability*, USA, 2005, pp. 26–36.

[9] K. Paulsen, "SATA, SAS, and RAID", in *Moving Media Storage Technologies*, Elsevier, 2011, pp. 125–163.

[10] D. Loshin, "Big Data Tools and Techniques", in *Big Data Analytics*, Elsevier, 2013, pp. 61–72.

[11] B. Fechner, "Fast online error detection and correction with thread signature calculae", *Microprocessors and Microsystems*, vol.36, no.6, pp. 462–470, 2012.

[12] C. Liu, C. Yang, X. Zhang, J. Chen, "External integrity verification for outsourced big data in cloud and IoT: A big picture", *Future Generation Computer Systems*, vol. 49, pp. 58–67, 2015.

[13] R. Rivest, *The MD5 message-digest algorithm*, 1992. Available: https://tools.ietf.org/html/rfc1321.

[14] *SHA1 version 1.0*. (n.d.), Retrieved May 10, 2015

[15] S. R. Ellis, "Fundamentals of Cryptography", in *Cyber Security and IT Infrastructure Protection*, Elsevier, 2014, pp. 295–307.

[16] P. Gupta, S. Kumar, "A Comparative Analysis of SHA and MD5 Algorithm", *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4492-4495, 2014.

[17] M. Stevens, *On collisions for MD5*, Eindhoven University of Technology, 2007.

[18] I. Mironov, *Hash functions: Theory, attacks, and applications*, Microsoft Research, Silicon Valley Campus. Noviembre de, 2005.

[19] S. Gupta, N. Goyal, K. Aggarwal, "A Review of Comparative Study of MD5 and SSH Security Algorithm", *International Journal of Computer Applications*, vol. 104, no. 14, 2014.