ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 8, Issue 8 , August 2021

A Framework for Preserving the Privacy of Data in Hadoop Clusters using Column Encryption

Hidayath Ali Baig, Dr.Farhat Jummani, Syed Zakir Ali

Research Scholar, Department of Computer Science, Shri Jagdishprasad Jhabarmal Tibrewala University, Jhunjhunu, Rajasthan 333001, India
Assistant Professor, Department of Computer Science, Shri Jagdishprasad Jhabarmal Tibrewala University, Jhunjhunu, Rajasthan 333001, India
Professor, Department of Computer Science, Secab Institute of Engineering and Technology Vijayapur, Karnataka 586101, India

ABSTRACT: Hadoop is a widely used distributed computing platform for processing large amounts of data on a file system termed as Hadoop Distributed File System (HDFS). Hadoop can be used to build cloud computing systems with high throughput, fault tolerance, and data availability. As the number of cloud services expanded, data privacy became a major concern .Initially, the Hadoop security model proved ineffective, demonstrating the weakness of the Hadoop protection mechanism. It is important to encrypt HDFS for data protection. As a result, a more complete protection system for sensitive data is required. In this paper, efforts have been made to solve the privacy issues in big data storage. We present a novel framework for data-at-rest using columnar-data storage and encryption approaches in the Hadoop ecosystem to preserve privacy. We adopted an Optimized Row Column (ORC) file format for storage and suggested an encryption mechanism. We believe that with this novel approach, we can achieve privacy of Big data with improved performance by encrypting only sensitive columns. In our experiments, we used various datasets such as flight information and weather station data. The obtained results showed decreased data load time and significant improvement in query processing time using the proposed ORC format with the ZLIB compression method.

KEY WORDS: Big Data, Columnar-File Format, Compression, Encryption, Hadoop, HDFS, Privacy.

I.INTRODUCTION

A number of data analytics techniques are utilized to achieve various, complicated data extraction tasks. New analytical techniques have the potential to change the way companies learn and function. Retailers use big data analytics to predict the demand for a variety of products. This data is used to identify and foresee the demand for the products [1]. When it comes to big data, the process is more sophisticated when compared to traditional methods and strategies, which makes data analysis more complex and repetitive [2]. Hadoop adoption is growing rapidly, and companies are including extra security measures to protect their applications from attack. Big Data is so important that it has created numerous privacy issues in addition to all of the great customer advantages[3]. Personal ownership of information is the foundation of privacy legislation. However, it is not clear that restricting information collection is frequently, without a doubt, a viable answer to privacy. Efforts have been undertaken in recent years to develop techniques for securing Hadoop and providing anonymity.

HDFS provides end-to-end encryption. The files can be read and written transparently encrypted and decrypted until the file has been configured. Protection from end-to-end ensures that the authorized recipient can only decrypt the data. HDFS never retain or has connections to unencrypted data or data encryption keys. That meets two typical encryption demands: at-rest (i.e., data stored on permanent media such as a disk) encryption and in-transit (e.g., When data passes across the network) encryption [4]. The following table1 provides a brief comparison between encryption methods applied to data stored in the Hadoop ecosystem.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

Table 1: Encryption of Data in Hadoop [5]

Volume Encryption	Application-level Encryption	HDFS data-at-rest Encryption
Protects data after physical theft or accidental loss of a disk volume	Encryption within an application running on top of Hadoop	Encrypts selected files and directories stored ("at rest") in HDFS
Entire volume is encrypted: very coarse-grained security	Supports a higher level of granularity and prevents "rough admin" access	Uses specially designated HDFS directories knows as "Encrypted Zones"
Does not protect against virus or other attacks that occur while the system is running	Adds a layer of complexity to the application architecture	End-to-end encryption of data read from and written to HDFS. HDFS does not have to access to unencrypted data or keys.

However, the simple file-level permissions and access control methods are carried out from the early stage of design of the actual Hadoop protection systems. Encryption is a gateway to the reliability of Hadoop and a means of passing secure HDFS data during MapReduce activities [6]. For the stable HDFS, encryption can be extended to HDFS. [7], [8]. Many secure data systems incorporate encryption mechanisms such as cryptography to encrypt data inside the Hadoop cluster. Encryption may produce performance failure by encrypting all the information rather than encrypting sensitive data [9].

Therefore, in this paper, we present a novel method for preserving personal privacy by encrypting sensitive data using columnar-file storage format instead of encrypting entire data in HDFS. The columnar file format is compatible with Hadoop and provides desired results on Hadoop clusters [10].

The rest of this paper is structured as follows. Section 2 goes through the related work in depth. Section 3 clarifies the plan for data-at-rest privacy using the columnar-storage scheme, and Section 4 addresses the experimental findings. Finally, the paper concludes with Section 5.

II. RELATED WORK

Big data protection and safety are problems that need to be discussed by academics and researchers. Several initiatives are in development to safeguard data privacy and availability. Hu et al. introduced an Access Management Framework for Big Data [11], which describes the concepts of authentication agreements, control lists, authentication policies, and the trust chain to users that have been granted access. The study provides a closer look at access control in big data modelling systems, including many Hadoop integrations descriptions. However, the details of the Hadoop environment are not properly discussed in this paper. Cloudera has been a key player in providing stable Hadoop implementations and has included many critical protection enhancements in their patented Hadoop software solutions [12]. A comparison paper from Gupta et al. describes how numerous access restrictions may be enforced in Hadoop utilizing Apache Ranger [13]. Sharma et al. commented on the security conditions of Hadoop [14]. Zeng et al. suggested a principle of access management for content-dependent data exchange on the Internet [15]. A managed scheme for data security focused on the proxy re-encryption technique for distributing personal data in the cloud was addressed by [16]. Ulusoy et al. have identified essential access control mechanisms for MapReduce named GuardMR and Vigiles [17], [18], respectively. Colombo et al. published a report showcasing "Big Data Security Best Practices" [19]. Big data computing and privacy protection systems are discussed in [20]. There are several other important Big Data privacy-related works are explored in [21] [22], [23]. ABAC (attribute-based access control) has gained popularity, and several researchers have built models based on it. Jin et al. unified a three-layer architecture as ABAC, Discretionary Access Control (DAC), and Mandatory Access Control (MAC) [24]. The National Institute of Standards and Technology (NIST) has suggested some methods to be used in Role-based access control (RBAC) [25]. Some other related access control models [26], [27], [27],[31] have motivated and encouraged our engagement in this research.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

III. PROPOSED METHODOLOGY

This paper proposes a multi-step method for protecting sensitive data and improving Hadoop cluster efficiency utilizing columnar data storage formats. For data protection, we define sensitive columns using a basic comparison approach based on index values, and we describe the various degrees of sensitivity levels proposed by Rao and Satyanarayana [32]. For reducing the data load time on clusters, we identified the effective compression mechanism. We propose to apply encryption on identified sensitive columns, suggest a key management strategy and data masking techniques.

Columnar Data Format

In the recent past, column-oriented data stores have gained popularity over traditional data storage mechanisms [33]. Hadoop Optimized Row Column (ORC) is a columnar file format that offers optimized streaming reads and integrated support for locating necessary rows rapidly. In this format, the data store organizes data in a column-oriented layout.

A) ORC File Structure

The ORC files consist of a group of rows, along with auxiliary information in the file footer. Postscript contains encoding criteria and compressed footer scales at the end of the file. Large strip sizes allow high-performance in Hadoop. The file footer describes the file strips, the number of rows per strip, and the different data types in each column. In this approach, users can also obtain information aggregated into min, max, and total categories. The following figure 1 shows the arrangement of ORC files:

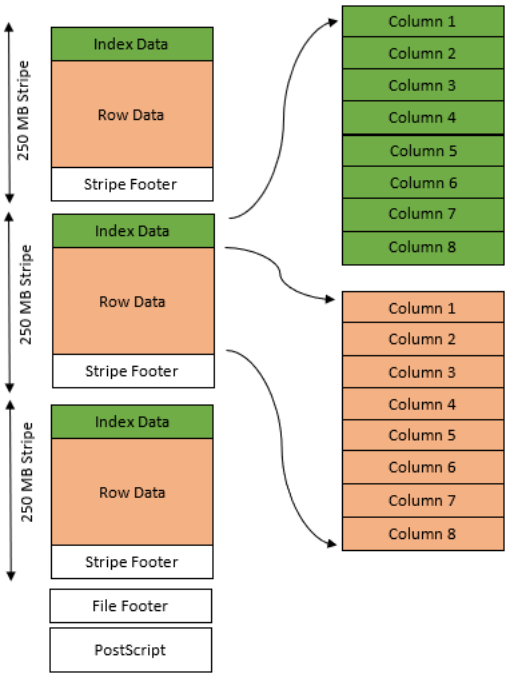

Fig 1.ORC file structure (Apache Software Foundation [10])

B) ORC Compression Method for HDFS

The ORC file uses various methods to achieve superior compression, including dictionary encoding, text delta encoding, run-length encoding, and particularly lightweight compression bitmap encoding. It aims at storage and its usefulness combined with better compression ratios [10].

Importing massive quantities of data needs high load reliability. Using any form of compression for transmitting data may improve efficiency while still providing privacy. In this research, different compression approaches such as ZLIB and SNAPPY are compared and presented in the results and discussion section to adopt an effective compression

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

technique on ORC file format. The following figure 2 shows the comparison of the file size of various encoding methods:

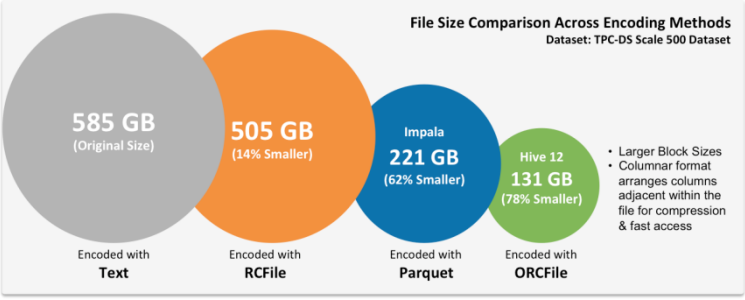

Fig 2. File Size Compression Across Encoding Methods

Encryption Mechanism

We propose to encrypt column details and statistics on disk to safeguard sensitive information. Column protection offers confidentiality even though many users have access to the file by applying fine-grained access control. For the recipient and writer, the encryption is transparent. The column encryption capability would be completely compliant with Hadoop Key Management Server (KMS) and will use the KMS to handle the column master keys. If the user has access to the keys, the original data is displayed. If a client does not have the key, masked information is produced.

Key Management Strategy

Each encrypted column will have a random local key created for it. As all decryption occurs locally, a user who stores the key can only access the column in that file. Keys are encrypted by the Hadoop or Ranger key management server (KMS). When ORC is configured to run with Ranger, it produces a random key [34]

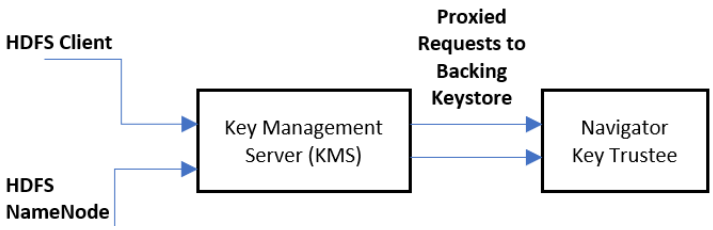

Fig 3. High-level design of Key Management Strategy. (Cloudera, Inc)

We propose to use Key Management Server (KMS) as a gateway between the cluster clients and the backup Keystore, revealing the Hadoop KeyProvider interface to the clients through the REST API. In addition to the local encrypted keys, the file often recognizes the master key used to encrypt the local keys.

Encryption Flow

This section highlights the proposed encryption flow for handling the key management process. The process starts by submitting the job to the cluster. The user's task then opens up the ORC file to read the encrypted key out of the ORC file. Then it will pass the encrypted key to the key management server; the decrypted key comes back if the user has permission, and the user can read the encrypted data out of the file. With this approach, the data does not flow through the KMS, but the user never gets the master key used to encrypt the local key.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

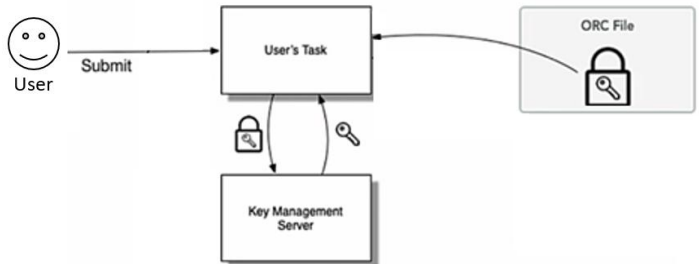

Fig 4. Encryption Flow

The local key is random, and for each file, it is managed by the user, and the file initialization vector is guaranteed to be unique.

Data Masking Strategy

When the user does not have access to the key, the data is defined with some static masks that correspond to Ranger's dynamic mass and the default masking strategy is nullifying all the values. If the user does not have access to sensitive information, null data is displayed to the user.

Proposed Data-at-rest Protection Pipeline

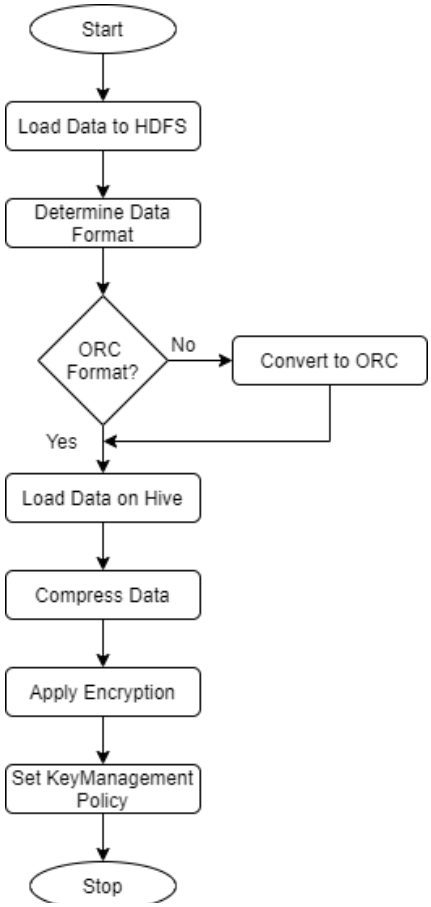

Fig 5. Proposed Data storage pipeline

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

ORC Data Loading Strategy

The recommended procedure is to load data as text into HDFS, construct an external Hive table over it, and then store the data in Hive as ORC, where it becomes a table handled by Hive.

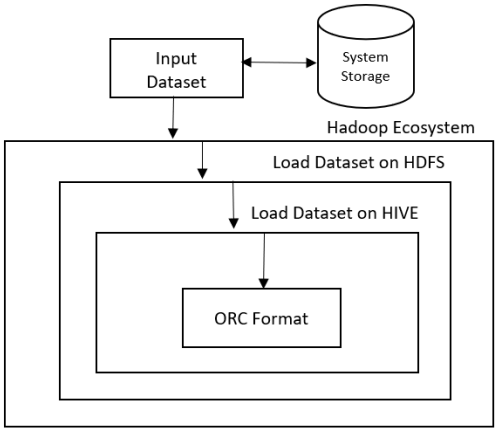

Fig 6. Hadoop-Hive Data Loading Strategy

IV. EXPERIMENTAL RESULTS

Our experiment tested the factors such as storage space occupied by different data formats, conversion time taken to convert existing data to ORC format, compression techniques, and time to retrieve the records.

Table 2: Comparison of Conversion and Retrieval Time between File Formats and Compression Techniques

	CSV	ORC-ZLIB	ORC-SNAPPY
Data Size in GB	11.5	1.1	1.6
Conversion time from CSV to ORC	NA	9.10	9.15
Number of Files	21	42	42
Time for fetching Single Record (Seconds)	255	70	62

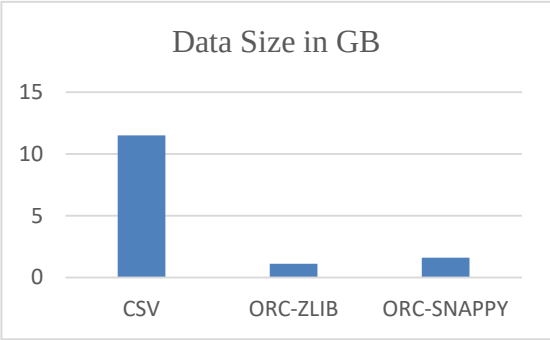

Fig 7.Data Size comparison

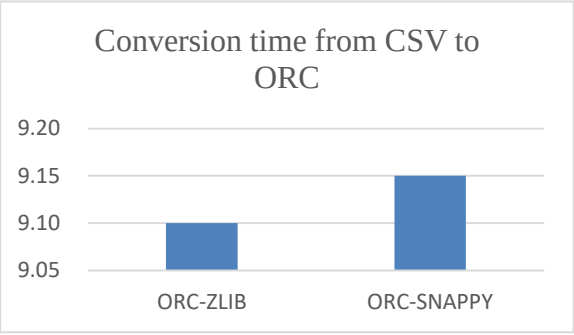

Fig 8. Data Conversion Time

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 8, Issue 8 , August 2021

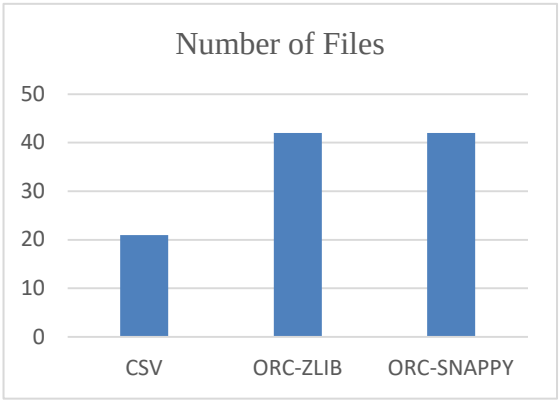

Fig 9. Number of Files

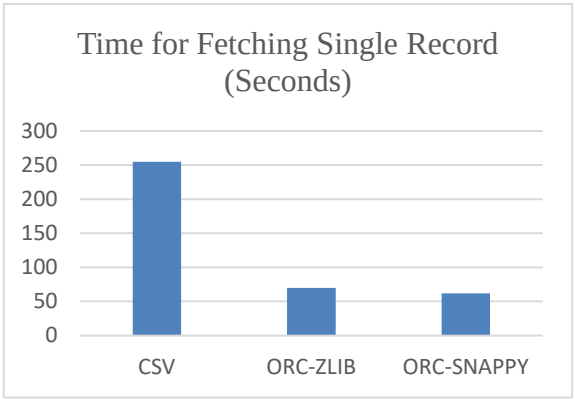

Fig 10. Record fetching time

The experiment findings show that ORC file format works better than CSV or any other text file format for maximizing storage capacity and improving query processing speed. ORC can consume less disk space relative to the text file format and optimize the loading time of the query, as ORC allows auto-partitioning and auto-indexing techniques [35]. Since there is a close comparison between ZLIB and SNAPPY compression for fetching the data, we further analyzed these compression techniques to identify a better solution. Research by Rattanaopas et al. found that the highest compression rate for the accuracy of data compression using ZLIB is 7:1 in comparison with NONE. However, the ratio of compression of SNAPPY was 4:1. All the outcome is inclined towards the same ratio across all rows of the tables [14]. On the weather station table, the scale of each ORC compression form is shown in Fig. 11 below.

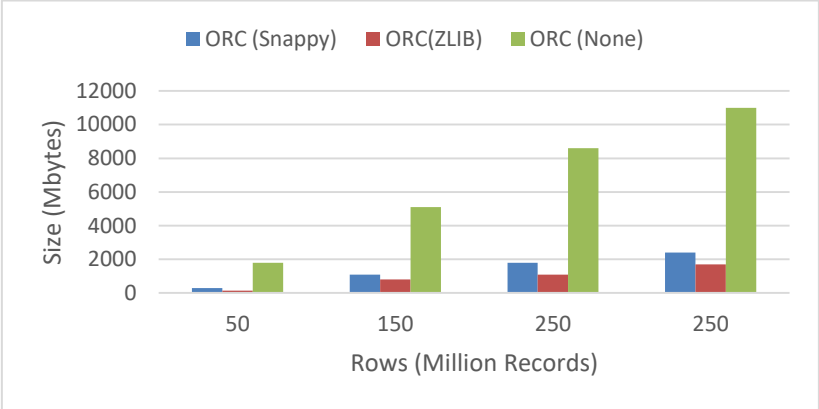

Fig 11. Size of the table on each ORC compression type (Rattanaopas et al., 2016 [14]).

Hive commands with separate compression algorithms such as Snappy and Zlib are performed in Hive. The result of the algorithms is provided in Table 3.

Table 3: ORC Compression ratio by Hive - (Rattanaopas et al., 2016 [14])

No.	Type	Input Data Size (Byte)	Compressed Data Size (Byte)	Compression Ratio
1	ORC + Snappy	9582763	3188612	0. 333
2	ORC + Zlib	9582763	1775361	0. 185
3	ORC + Zlib	628458887	40052331	0. 064

The ORC + Zlib is 55% of the ORC + Snappy. Hence the compression combination of ORC + Zlib is preferable. It indicates that the compression ratio for small ORC + Zlib files is far better for larger files (6.4%).

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 8, Issue 8 , August 2021

V.CONCLUSION AND FUTURE WORK

In conclusion, the proposed columnar storage model is applied to protect sensitive data for better storage and lightweight encryption. Encrypting only a subset of columns is effective than encrypting the whole file or table. If a database just accesses a few columns of a table, I/O may be greatly decreased versus conventional row-oriented storage. The data type recognition and uniformity can be used to improve the performance of encryption and compression algorithms. We also discussed the encryption mechanism, encryption key management methods, and data masking. Our results prove that in comparison to storing data in text file and ORC file format, ORC format produces effective data storage in Hadoop, improving efficiency with no data loss. Using ORC with ZLIB compression boosts performance. The limitations of the proposed approach are the need for a proper encryption policy for writing data. If any changes to the masking policy require re-writing files, it also requires additional data masks for critical data like credit cards or address information. Decrypted local keys could be saved locally in client machines which in turn increases the security risk. Hence, further work is required to provide standard encryption and data masking policies and improved key management strategies to address these challenges.

REFERENCES

[1]Rebecca Herold, "10 Big Data Analytics Privacy Problems," Jun. 26, 2014. https://privacysecuritybrainiacs.com/privacy-professor-blog/10-big-data-analytics-privacy-problems/.
[2]D. Y. K. Sharma and S. V. G. Sridevi, "Using Big Data Analytics in order to Understand and Take Care of Environmental Emergencies," vol. 6, no. 6, p. 6, 2019.
[3]D. Y. K. Sharma, V. Pujari, and R. Rane, "A REVIEW PAPER ON BIG DATA AND HADOOP," vol. 7, no. 1, p. 5, 2020.
[4]Apache Software Foundation, "Hadoop," Jan. 10, 2020. hadoop.apache.org.
[5]Apache Software Foundation, "Hortonworks Data Platform," 2021. https://www.cloudera.com/products/hdp.html (accessed Jan. 09, 2021).
[6]S. Park and Y. Lee, "Secure Hadoop with Encrypted HDFS," in Grid and Pervasive Computing, vol. 7861, J. J. Park, H. R. Arabnia, C. Kim, W. Shi, and J.-M. Gil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 134–141.
[7]Q. Shen, L. Zhang, X. Yang, Y. Yang, Z. Wu, and Y. Zhang, "SecDM: Securing Data Migration between Cloud Storage Systems," in 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia, Dec. 2011, pp. 636–641.
[8]H.-Y. Lin, S.-T. Shen, W.-G. Tzeng, and B.-S. P. Lin, "Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System," in 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, Mar. 2012, pp. 740–747.
[9]M. M. Shetty and D. H. Manjaiah, "Data security in Hadoop distributed file system," in 2016 International Conference on Emerging Technological Trends (ICETT), Kollam, India, Oct. 2016, pp. 1–5.
[10]Apache Software Foundation, "ORC Specification v1," 2021. https://orc.apache.org/specification/ORCv1/.
[11]V. Hu, T. Grance, D. Ferraiolo, and D. Kuhn, "An Access Control Scheme for Big Data Processing," presented at the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Miami, United States, 2014.
[12]Cloudera, Inc., "Apache Hadoop Ecosystem," 2020. https://www.cloudera.com/products/open-source/apache-hadoop.html.
[13]M. Gupta, F. Patwa, and R. Sandhu, "Object-Tagged RBAC Model for the Hadoop Ecosystem," in Data and Applications Security and Privacy XXXI, vol. 10359, G. Livraga and S. Zhu, Eds. Cham: Springer International Publishing, 2017, pp. 63–81.
[14]K. Rattanaopas, S. Kaewkeerat, and Y. Chuchuen, "A Comparison of ORC-Compress Performance with Big Data Workload on Virtualization," Appl. Mech. Mater., vol. 855, pp. 153–158, Oct. 2016.
[15]W. Zeng, Y. Yang, and B. Luo, "Access control for big data using data content," in 2013 IEEE International Conference on Big Data, Silicon Valley, CA, USA, Oct. 2013, pp. 45–47.
[16]D. Nunez, I. Agudo, and J. Lopez, "Delegated Access for Hadoop Clusters in the Cloud," in 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, Singapore, Dec. 2014, pp. 374–379.
[17]H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, and E. Pattuk, "GuardMR: Fine-grained Security Policy Enforcement for MapReduce Systems," in Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15, Singapore, Republic of Singapore, 2015, pp. 285–296.
[18]H. Ulusoy, M. Kantarcioglu, E. Pattuk, and K. Hamlen, "Vigiles: Fine-Grained Access Control for MapReduce Systems," in 2014 IEEE International Congress on Big Data, Anchorage, AK, USA, Jun. 2014, pp. 40–47.
[19]P. Colombo and E. Ferrari, "Privacy Aware Access Control for Big Data: A Research Roadmap," Big Data Res., vol. 2, no. 4, pp. 145–154, Dec. 2015.
[20]R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," IEEE Netw., vol. 28, no. 4, pp. 46–50, Jul. 2014.
[21]H. A. Baig, Dr. Y. K. Sharma, and S. Z. Ali, "Privacy-Preserving in Big Data Analytics: State of the Art," SSRN Electron. J., 2020.
[22]J. Soria-Comas and J. Domingo-Ferrer, "Big Data Privacy: Challenges to Privacy Principles and Models," Data Sci. Eng., vol. 1, no. 1, pp. 21–28, Mar. 2016.
[23]O. Tene and J. Polonetsky, "Big Data for All: Privacy and User Control in the Age of Analytics," p. 38.
[24]X. Jin, R. Krishnan, and R. Sandhu, "A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC," in Data and Applications Security and Privacy XXVI, vol. 7371, N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–55.
[25]D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding Attributes to Role-Based Access Control," Computer, vol. 43, no. 6, pp. 79–81, Jun. 2010.

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 8, Issue 8 , August 2021

[26]M. A. Al-Kahtani and R. Sandhu, "A model for attribute-based user-role assignment," in 18th Annual Computer Security Applications Conference, 2002. Proceedings., Las Vegas, NV, USA, 2002, pp. 353–362.
[27]R. S. Sandhu, E. J. Cope, H. L. Feinstein, and C. E. Youman, "RolemBased Access Control Models," p. 10, 1996.
[28]S. Bhatt, F. Patwa, and R. Sandhu, "ABAC with Group Attributes and Attribute Hierarchies Utilizing the Policy Machine," in Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control - ABAC '17, Scottsdale, Arizona, USA, 2017, pp. 17–28.
[29]J. Crampton and G. Loizou, "Administrative scope: A foundation for role-based administrative models," ACM Trans. Inf. Syst. Secur., vol. 6, no. 2, pp. 201–231, May 2003.
[30]M. Gupta and R. Sandhu, "The GURAG Administrative Model for User and Group Attribute Assignment," p. 15, 2016.
[31]X. Jin, R. Sandhu, and R. Krishnan, "RABAC: Role-Centric Attribute-Based Access Control," in Computer Network Security, vol. 7531, I. Kotenko and V. Skormin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 84–96.
[32]P. S. Rao and S. Satyanarayana, "Privacy preserving data publishing based on sensitivity in context of Big Data using Hive," J. Big Data, vol. 5, no. 1, p. 20, Dec. 2018.
[33]A. Floratou, J. M. Patel, E. J. Shekita, and S. Tata, "Column-oriented storage techniques for MapReduce," Proc. VLDB Endow., vol. 4, no. 7, pp. 419–429, Apr. 2011.
[34]Apache Software Foundation, "Transparent Encryption in HDFS," Jan. 10, 2021. https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/TransparentEncryption.html.
[35]R. Kumar and N. Kumar, "Improved join operations using ORC in HIVE," CSI Trans. ICT, vol. 4, no. 2–4, pp. 209–215, Dec. 2016.