2023 8th International Conference on Communication and Electronics Systems (ICCES) | 979-8-3503-9663-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCES57224.2023.10192702

Innovative Approach of Data Encryption Algorithm for Securing Big Data

Shivani Awasthi
Research Scholar, Department of Computer Science
Harcourt Butler Technical University, Kanpur, India
shivanitiwari2184@gmail.com

Prof Narendra Kohli
Professor, Department of Computer Science
Harcourt Butler Technical University, Kanpur, India
nkohli@hbtu.ac.in

***Abstract*: Nowadays, a large amount of data that is growing exponentially is known as Big data. In other words, big data contains more variety and comes in big volume with more velocity. Traditional processing software cannot handle complex and big amounts of data, but this data addresses a lot of business problems, that could not be able to tackle before so to deal with this data, open-source software HDFS (Hadoop Distributed File System) is used as a cloud storage system. Big Data Security is a collective term used for all measures and tools that are used to guard data and analytics processes used for accessing the data. Big data is a joint term and tool for data and data management systems against attacks, thefts, or other malicious activities that could harm or negatively affect them so server and serverless systems both are major concerned with high security of data. The author's research presents data security in the cloud computing environment in this paper. HDFS does not provide any scheme of encryption and decryption for securing the data, HDFS act as a storage medium that stores data in Avro format. Users use the encryption key and encryption zone to encrypt their data into HDFS.**

Keywords*: Big Data, Encryption Zone, Encryption Key, Hadoop, HDFS (Hadoop Distributed File System), Key Management Server (KMS), Transparent Data Encryption (TDE).

I. INTRODUCTION

Recently, Big Data is considered as an emerging and challenging field. Big Data is growing exponentially with time. In other words, Big data definition by the IDC: "Big Data technologies describe a new generation of technologies and architectures, designed to economically extract value from the large volumes of a wide variety of the data, by enabling high-velocity capture, discovery, and/or analysis" [1]. Cloud infrastructure is used for the Big data application as an ideal platform. High flexibility, enabling various services automatically, scalability, and fault tolerance are the features of Cloud platforms, so users use those features to make high-performance clusters with many preconfigured services for Big data platforms [2] [3]. The security of Big data is a major concern when the number of applications on the cloud is increasing day by day. In this era of cloud computing environment, there are four control measures subject such as flow control, data control, inference control, and encryption. In a Distributed system, Database security is needed, an encrypted database on the hard disk and any backup media is the new approach for protecting sensitive data from theft and fraud. In this way, we can say that Business perspective, data protection is a major issue in every sector like healthcare, financial, and government sector.

Hadoop is a recent technology that is used for cloud storage. Hadoop stores large data and processes those large data sets efficiently. It is an open-source framework dealing with large data sets from gigabytes to petabytes [3][4]. Hadoop allows clustering many computers to analyze parallelly massive datasets more quickly Instead of using one large computer to process and store the data [4]. Hadoop is designed without concern for the security point so only stores data and runs the job parallelly with large data sets. In this case, we use Transparent Encryption for Hadoop [5] run on the Cloudera platform [6]. Cloudera's "Security of Hadoop" follows the important steps for providing the security of data on the Hadoop framework for running Big data applications [6]. In securing the data, a protection technique is used based on the encryption method. In a Cloud environment high accessibility, multiple copies of data, and security of data is the most important point, so an end-to-end encryption approach is used by many organizations for securing large data and for analytical processing. End-to-end encryption is controlled by the organization only [7]. HDFS contains transparent end-to-end encryption once configured the data is read and written from special HDFS directories without any change to the user application code. This means data is encrypted and decrypted by the client. HDFS never stores unencrypted data and unencrypted keys.

The remainder of this paper is ordered as follows: section II describes the framework of Hadoop with HDFS architecture (Hadoop Distributed File System) section III discussed security, section IV discussed the encryption of data in detail, and Section V attack vectors and the last section VI conclusion of this paper.

II. FRAMEWORK OF HADOOP

Hadoop [8] [9] is a distributed platform as well as efficient and reliable. It is an open-source software from Apache, which is based on Java language. Hadoop provides scalable storage and processes large data sets. Due to its features, it is widely accepted by business organizations and many industries. Many organizations such as Twitter, Yahoo, and Facebook used the Hadoop platform for dealing with large datasets in a scale-out (Horizontal scaling concept) manner. In dealing with large data sets two techniques are used scale-out and scale-in. In a cloud computing environment, scale-out techniques are used for dealing with large data sets [10] [11]. Hadoop has two units, map reduce and HDFS [8] [9]. Hadoop was created by Google based on the Google File System and map-reduce papers [12] [13] published by Google in 2003 and 2004 years. Map-reduce is used for processing a large amount of data sets parallelly. HDFS is used to store large data sets in distributed clusters of machines [12] [13].

High availability and fault tolerance point of view, files are stored in replication patterns across multiple clusters over the network. Distributed scalable computing framework is the most trending technique now-a-day for dealing the large data as well as for analytical processing. It provides a crystal-clear interface just like master-slave architecture for handling large data in multiple clusters [14]. Users interact with the name node to manage the stored file and to keep the metadata information of files in HDFS. The data node is used for storing the files that are stored in HDFS. Single to thousands of nodes manage by each cluster for storing as well as the parallel processing of large data sets.

A. Architecture of Hadoop

Hadoop's main service is, running on the low-cost commodity of hardware with high availability and fault tolerance capability for access to applications with high throughput. The HDFS architecture is shown in Fig 1. An HDFS cluster has two nodes, name node is one, for managing the files by maintaining the hierarchy of files as well as metadata information of the files via the backup node and the other is the data node stores the data of the file in blocks.

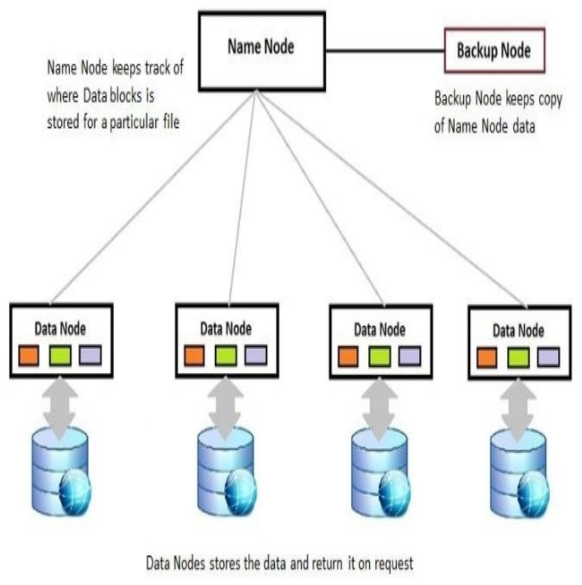
Name Node
Backup Node
Name Node keeps track of where Data blocks is stored for a particular file
Backup Node keeps copy of Name Node data
Data Node
Data Node
Data Node
Data Node
Data Nodes stores the data and return it on request

Fig. 1. (https://www.linkedin.com/pulse/analysis-hadoop-hdfs-distributed-file-system-amit-kriplani)

If the user wants to store the data, the user interacts with the name node. HDFS breaks the data of the file into smaller blocks. This process is known as input splits and a block of data is kept across several nodes throughout the cluster and keeps track of it. For high availability and fault tolerance-based features, it ensures the availability of data by keeping multiple copies of it in different data, nodes based on Hadoop 3. x version. In another case when the user wants to read the data, the user interacts with the name node to get file information from the metadata and then communicates to the data node. The name node contains all the information such as the file location in different data nodes based on metadata information. when the user wants to write data in the data node, the user interacts with the name node, name node provides the information about the location where the user kept the data in the data node and kept this information for future reference and then the user writes the data in a particular location. Avro File format has efficient storage and wide support inside and outside the Hadoop ecosystem. This format is ideal for long-term support for storing important data. This format also read and writes in many languages such as Scala and Java.

III. SECURITY REQUIREMENT

Design of the Hadoop framework, the main attention was on storage and then processing large data set parallelly, so security issues is missed when the Hadoop framework was designed by Apache. In the initial phase of using HDFS to store the data of the different users, we use a single machine to store huge data and provide the service as a public cloud [14] [15]. In this case, the intruder accesses the data or stole data from the data node by accessing the login credential [16]. This is the weak point for the enterprise to analyze huge sensitive data then many organizations focus on the protection of sensitive data via end-to-end encryption [17]. In encryption, we focus on data at REST and data in transit [18]. In disk encryption, the main attention is on the encryption of hard drives to protect data from attackers. In Transparent data encryption (TDE), end-to-end encryption occurs so stored data in HDFS never encrypted or decrypted. It means the user only encrypted and decrypted on the client side.

Many organizations such as businesses and governments protect data via encryption and other techniques for example card payment using the PCI DSS (Payment Card Industry Security Standards), the healthcare sector using content cloud platforms to secure health sector data, protecting the sensitive data WhatsApp using AES-256 in CBC mode for encryption and HMAC-SHA256 for the authentication and Facebook uses SHA-256 for the encryption of data that are stored in the Hadoop environment. In the financial sector, HDFS stores many files across multiple machines in large clusters. Each file is broken into a sequence of blocks each block size is fixed, except the last block. Blocks are replicated across multiple nodes in clusters for fault tolerance. The block size and replicator factor are configured per file. HDFS used as storage and Hadoop performs the calculation locally and uses network bandwidth for only transmitting the result, in this way, we save the bandwidth of the network for transferring the whole data on the client side. In the above case, data protection is an important issue that is handled by the encryption technique. In this era, we require secure data in every phase of its life cycle from the loading to the processing of data through HDFS, which means firstly loading the data into HDFS, saving that data in the data node via encryption then operating on saved data and last the transfer the data from HDFS to the local File system. HDFS itself does not provide the protection of the data its plays as a storage space where the data is kept then protection is needed in a distributed cloud environment. A secure cluster, based on the encryption of sensitive data is the main point in a real-world scenario so implementing all the features in every phase of the life cycle of data is the main point in this era.

IV. SECURITY OF DATA AT REST

One of the solutions is encryption for the secure sensitive data so an intruder does not steal the data that reside in encrypted form in the Hadoop ecosystem environment. In the distributed environment, the main area of encryption of data is "data at REST" and "data at motion" or transit [18]. Data at REST means data stored data in encrypted form on the disks. Data in motion means data travel in encrypted form while it is moving.

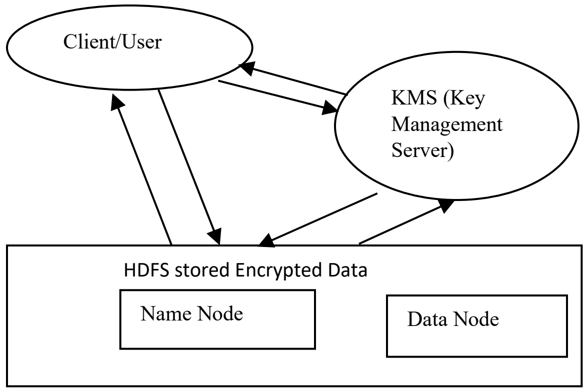

Fig.2. The Encryption process overview

This is transparent encryption, so the user enters the password and accesses sensitive data. A key management server is required for managing the keys. Sensitive data is firstly encrypted via symmetric and asymmetric keys and then loaded to the HDFS. In this case, HDFS contains encrypted data by the write operation and does not contain unencrypted data. The user while reading read-only encrypted data from the HDFS then the client decrypts the data [18] [19]. Fig. 2. Shows the encryption process overview. From the above Fig. 2. The Client request to store data and interact with KMS then KMS provides the key and also allows access to the HDFS then the client reads and writes HDFS via the name node through the data node.

In end-to-end transparent encryption [5] firstly configuration process in Cloudera is completed then the reading and writing of data are possible from the HDFS directories only the client encrypts and decrypts the data. Hadoop never accesses unencrypted data and encryption keys. Encrypted data is stored in HDFS and move through the network.

A. HDFS Architecture

HDFS transparent encryption new document introduced by Apache. In the Apache document for the Cloudera platform, we use the encryption zone [5] [6]. In the encryption zone, data is written in the encrypted form and read outside this zone the data is decrypted via the encrypted zone key that is created by the user. Each file in the encryption zone has a unique data encryption key (DEK) and encrypted DEK with an encryption zone key. This is not handled directly by

HDFS. Only EDEK (Encrypted Data Encryption Key) is handled by the HDFS. The client decrypts EDEK (Encrypted Data Encryption Key) and uses the DEK (Data Encryption Key) to read the data on the client side. The name node stores EDEK, and clients use this EDEK for reading and writing the data into the encryption zone [5] [6]. The data node stored only the encrypted data via streaming of encrypted byte form. For end-to-end encryption, we use KMS (Key Management Server) for communication between the client, HDFS, and key server based on Fig. 2. The Hadoop file system basically performs three duties.

1. Accessing the Encryption zone key
2. Decrypt EDEKs by the client
3. Generating the new EDEK for the name node

B. KMS (Key management server), Key provider, and EDEK

KMS is known as a proxy, that interfaces with a backing key store on behalf of HDFS clients and daemons. KMS and backing key stores implement APIs in Hadoop Key Provider. Key provider API has a unique key name due to multiple versions of the key and keys are rolled back. The key name and its version-based encryption key are fetched [5] [6].

Only KMS has enabled functionality for creating and decrypting the encrypted encryption key (EEK). It is important to note that an EEK's encryption key is never handled by the client that requests its creation or decryption. The fresh random key is generated by KMS and this key is encrypted with the chosen key and then sends the new EEK back to the client. Before decrypting the EEK and returning the decrypted encryption key, the user's possession of the encryption key is verified by the KMS. Using this method, an EEK is decrypted [5] [6].

EEK is an encrypted data encryption key in HDFS encryption, whereas a data encryption key (DEK) is used to encrypt and decrypt file data. The key store is often set up such that only clients have access to the keys for encrypting DEKs. As a result, HDFS managed and stored securely EDEKs and HDFS users will not have access to the EDEKs' unencrypted encryption keys.

C. workflow

HDFS client writes the new file and then follows the order [18]:

1. HDFS client call create () for writing the new file.
2. Create a new EDEK by EZK-id/version name node request KMS.
3. New DEK generated by the KMS and key server provides EZK.
4. DEK encrypts by the KMS, and EDEK sends to the name node.
5. Name node kept this EEDEK for file metadata.
6. EDEK provides to the client by the name node.
7. EDEK provided by the HDFS client to the KMS.
8. Requesting key server, KMS decrypts the EDEK using EZK.
9. HDFS client provides DEK.
10. HDFS client using DEK encrypts data and writes encrypted data block into HDFS.

Reading the encrypted file follow the sequence [18]:

1. Reading the file open () call by the HDFS client.
2. EDEK provided by the name node to the client.
3. HDFS client passes EDEK and EZK-id/version to the KMS.
4. KMS requests a key server for EZK and EZK-based decrypts EDEK
5. DEK provided to the client by KMS, and the client read an encrypted block of data and decrypted by the DEK.

All the steps are followed by the DFS client, name node, and KMS for reading and writing the file from the read-and-write path. This operation is fully controlled by the HDFS file system permission. If HDFS compromised (unauthorized access to the superuser account) then the attacker only accesses the encrypted data and encrypted keys. Encryption zone key accessing is controlled by a distinct set of permission on the key store and KMS that cannot breach the security. Fig 3. shows the overall process of encrypting and decrypting the files.

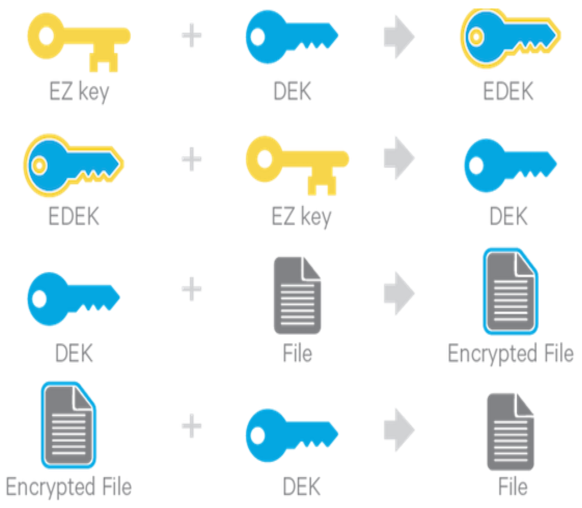

Fig. 3. Encrypt and Decrypt the file based on the EZ key (Encryption Zone Key), DEK key (Data encrypt keys), and EDEK key (Encrypted data encryption key) (https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/cdh_sg_hdfs_encryption.html)

KMS isolated from the Hadoop ecosystem to provide security so communication between HDFS client KMS and key server by the TLS.

V. ATTACK VECTORS

A. Hardware Access Exploits

In this type of attack, the attacker has access to the hard drives of the data nodes and name nodes in the cluster.

1. access to processes swap files that include data encryption keys. This is not sufficient to obtain cleartext by itself because it also needed access to encrypted block files. This can be decreased by disabling swaps, using encrypted swaps, or using mlock to prevent keys from being switched out.

2. access to encrypted files Since it also needs access to DEKs, this does not obtain cleartext on its own. This is solved by restricting physical access to cluster machines.

B. Root Access Exploits

These exploits presumptively work on data nodes and name nodes in a cluster of computers when the attacker has got root shell access. Malicious root users access the in-memory state of processes containing encryption keys and cleartext and many of these attacks cannot be fixed in HDFS. The sole defense against these exploits is to tightly limit and watch over root shell access.

1. encrypted block files accessing

By itself, without an encryption key does not obtain cleartext.

2. To obtain DEKs, delegation tokens, and clear text it requires dump memory of client processes.

No mitigation

3. To obtain encryption keys and encrypted data in transit, record network traffic.

By itself, without EDEK and encryption key, there is insufficient to read cleartext.

4. Dump memory of data node process encrypted block data obtained.

By itself, without DEK it is not possible to read cleartext.

5. Dump memory of the name node process to get encrypted data encryption keys.

By itself, without EDEK's encryption key and encrypted block file impossible to read cleartext.

C. HDFS Access Exploits

These vulnerabilities presuppose that HDFS compromised by the attacker but lacks access to root or HDFS user shells.

1. Encrypted block file access
By itself, without EDEK and encrypted EDEK lacks to read cleartext.
2 -fetchImage access encryption zone and encrypted file metadata
By itself, without EDEK and encryption key not possible to read cleartext.

D. Rogue User Exploits

A malicious person can gather keys that use in the future for encryption. With periodic roll key regulations policies, this issue can be solved.

VI. CONCLUSION

In the recent era of Big Data, all the data is compiled from various sources into a single distributed platform. In the above case, security plays the main role. This paper points out that storage-level encryption Kerberos fails in the protection of a large amount of data. In this study, we cover the security of sensitive data by the method of encryption to guard against malicious attacks in the Hadoop cluster. We discussed the recent addition of native data-at-rest encryption in HDFS. Performance degradation occurs when encryption takes place so in the future, we consider other techniques to encrypt only important data rather than all the data in the file that are stored in HDFS. In the Future, there is a need for the next range of Hadoop with a wide range of security techniques for protecting data and securing the execution of jobs parallelly inside the framework.

REFERENCES

[1] Gantz J, Reinsel D. Extracting value from chaos [J]. IDC view, 2011: 1-12.
[2] IBM Big Insights on Cloud, IBM, Accessed July 26, 2016. http://www-03.ibm.com/software/products/en/ibm-biginsights-oncloud.
[3] Amazon Elastic MapReduce (EMR), Accessed July 26, 2016. https://aws.amazon.com/emr
[4] https://geeksforgeeks.org
[5] "Transparent Encryption in HDFS." Apache Hadoop 2.7.2 –. Accessed July 26, 2016. https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoophdfs/TransparentEncryption.html.
[6] "HDFS Data At Rest Encryption". 2016. Cloudera.Com. Accessed July 26, 2016. https://www.cloudera.com/documentation/enterprise/5-4-x/topics/cdh_sg_hdfs_encryption.html.

[7] Zerfos, Petros, Hangu Yeo, Brent D. Paulovicks, and Vadim Sheinin. "SDFS: Secure distributed file system for data-at-rest security for Hadoop-as-a-service." In Big Data (Big Data), 2015 IEEE International Conference on, pp. 1262-1271. IEEE, 2015.

[8] White T.: Hadoop: The Definitive Guide, 1st ed. O'Reilly Media (2009)

[9] Hadoop, http://hadoop.apache.org/

[10] www.dell.com

[11] www.techtarget.com

[12] Dean J., Ghemawat S.: MapReduce: Simplified Data Processing on Large Cluster, In OSDI (2004)

[13] Ghemawat S., Gobi off H., Leung, S.: The Google File System. In: ACM Symposium on Operating Systems Principles (October 2003)

[14] Cheng, Zhonghan, Diming Zhang, Hao Huang, and Zhenjiang Qian. "Design and Implementation of Data Encryption in Cloud based on HDFS." In International Workshop on Cloud Computing and Information Security (CCIS 2013), pp. 274-277. 2013.

[15] Shehzad, Danish, Zakir Khan, Hasan Dag, and Zeki Bozkus. "A Novel Hybrid Encryption Scheme to Ensure Hadoop-Based Cloud Data Security. "International Journal of Computer Science and Information Security 14, no. 4 (2016): 480.

[16] Sudesh Narayanan, "Securing Hadoop", Packet Publishing, ISBN:9781783285266, 2013.

[17] Sooyoung Park and Young Seok Lee, Secure Hadoop with Encrypted HDFS, Springer-Verlag Berlin Heidelberg in 2013

[18] Spivey, Ben, and Joey Echeverria. Hadoop Security: Protecting Your Big Data Platform. Sebastopol, CA: O'Reilly, 2015.

[19] Lakhe, Bhushan. Practical Hadoop Security. A press, 2014.

[20] H. Lin, S. Shen, W. Tzeng, and B. P. Lin, Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System [C]// 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, 2012, pp. 740-747. Doi 10. 1109/ AINA. 2012. 28.

[21] Chao Yang, Lin Weiwei, Mingqi Liu. A Novel Triple Encryption Scheme for Hadoop-Based Cloud Data Security [C]// Emerging Intelligent Data and Web Technologies (EIDWT) 2013 Fourth International Conference on, pp. 437-442, 2013.

[22] http://github.com/intelha-doop/projecthino/

[23] ZHOU Dao Ming, QIAN Lufeng, WANG Lulu. Transparent encryption technology research [J]. Net info Security, 2011, 12 (14): 54-56. (in Chinese)

[24] https://www.projectpro.io/article/hadoop-in-financial-sector/80

[25] www.simplilearn.com7