International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 06 | June-2018 www.irjet.net p-ISSN: 2395-0072

SECURED HADOOP ENVIRONMENT

Safeeda[1], Prof. Savitha C. K[2], Prof. Prajna M. R[3], Prof. Ujwal U[4]

[1]M.tech Student, Dept. of Computer Science, K.V.G College of engineering, Sullia, Karnataka, India
[2,3]Professor, Dept. of Computer Science, K.V.G College of engineering, Sullia, Karnataka, India
[4]Head of the Department, Department of Computer Science, K.V.G College of Engineering, Sullia, Karnataka, India

---

Abstract-*Data is growing at an enormous rate in the present world. One of the finest and most popular technologies available for handling and processing that enormous amount of data is the Hadoop ecosystem. Enterprises are increasingly relying on Hadoop for storing their valuable data and processing it. However, Hadoop is still evolving. There is much vulnerability found in Hadoop, which can question the security of the sensitive information that enterprises are storing on it. In this paper, security issues associated with the framework have been identified. We have also tried to give a brief overview of the currently available solutions and what are their limitations. At the end a novel method is introduced, which can be used to eliminate the found vulnerabilities in the framework. In the modern era, information security has become a fundamental necessity for each and every individual. However, not everyone can afford the specialized distributions provided by different vendors to their Hadoop cluster. This paper presents a cost-effective technique that anyone can use with their Hadoop cluster to give it high security.*

Keywords: Big data, encryption, Hadoop, information security, Mahout, etc…

1. INTRODUCTION

Big Data is the term that refers not only to the large volumesof data but it is also concerned about the complexity of the data and the speed at which it is getting generated. It is generally described by using three characteristics,widely known as 3 V's:

- **Volume**

The size is one of the characteristics that define big data. Big data consists of very large data sets. However, it should be noted that it is not the only one parameter and for data to be considered as big data, other characteristics must also be evaluated.

- **Velocity**

The speed at which the data is being generated is an important factor. For example, in every one second thousands of tweets are tweeted on micro blogging platform, Twitter. Even if the size of each individual tweet is 140 characters, the speed at which it is getting generated makes it an eligible data set that can be considered as big data.

- **Variety**

Big data comprises data in all formats: structured, unstructured or combination of both. Generally, it consists of data sets, so complex that traditional data processing applications are not sufficient to deal with them. All these characteristics make it difficult for storing and processing big data using traditional data processing application software's. Two papers published by Google [1], [2] build the genesis for Hadoop. Hadoop is an open source framework used for distributed storage and parallel processing on big data sets.

Two core components of Hadoop are:

- **Hadoop distributed file system (hdfs)**

Used for distributed storage of data. The input file is first split into blocks of equal size except the last block which are then replicated across Data Nodes. Currently, default block size is 128 MB which was previously 64 MB and default replication factor is 3. Block size and replication factors are configurable parameters.

- **Mapreduce**

It is used for parallel processing on distributed data on cluster of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data set intoindependent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the map function, which are then given as input to the reduce tasks. There are plenty of resources available [3], [5] that describe the detailed architecture of Hadoop and about how it works.

Our focus in this paper is on identifying the security problems that can occur on a typical Hadoop environment, what are the currently available solutions for encrypting data in Hadoop and their limitations, and finally, a novel approach that can be used for large scale encryption in Hadoop environment in efficient manner will be discussed.

2. HADOOP SECURITY

To perceive a basic idea about Hadoop security, one has to look back at the origin of Hadoop. Security was not exactly the priority for Doug Cutting and his team at the time they started developing the Hadoop [6]. Initially, it was just a part of Apache Nutch project until it was moved to the new Hadoop subproject in January 2006 [7]. A basic authentication level using username and password is provided in default Hadoop installation. Many a times, it will not be sufficient. As a default, Hadoop doesn't give support for any authentication of its users or Hadoop services. A user only authenticates with the operating system during the logon process. After this process, when the user invokes the Hadoop command, the user id and group is set by executing whoami and bash -c groups respectively. So if a user writes their own whoami script and adds it to the path before the Linux whoami is called, the user should beable to impersonate any user including the super user in the Hadoop file system.

- **Security of data in motion**

In Hadoop, actual data is stored in distributed manner on multiple DataNodes while NameNode stores the metadata and edit log. The actual communication of data blocks happens between Client Node and Data Node. Hence, Hadoop framework consists of several nodes with data communication between them. The data is transmitted over the network, which is not encrypted by default. The data which is being transmitted is therefore open to attack by hackers. Various communication protocols such as Remote Procedure Call (RPC), Transmission Control Protocol over Internet Protocol (TCP/IP), and Hypertext Transfer Protocol (HTTP) are used for internode communication. There are some solutions available for securing communication between various nodes like Kerberos and Simple Authentication and Security Layer (SASL) which are discussed in next section.

- **Security of data at rest**

Data at rest refers to data stored in persistent storage. By default Hadoop doesn't encrypt data that's stored on disk and that can expose sensitive data to security attacks. This is especially a big issue due to the nature of Hadoop architecture, which spreads data across a large number of nodes; exposing the data blocks at all those unsecured entry points. There are a number of choices for implementing encryption at rest with Hadoop. One of them is to use encryption zone a new layer of abstraction to HDFS. It will be discussed along with its limitations in next section.

We have to also understand that since Hadoop usually deals with large volumes of data and encryption/decryption takes time, it is important that the framework used performs the encryption/decryption fast enough, so that it doesn't impact performance.

Today, Hadoop is no more a tool for experimentation. Rather, it is getting increasing popularity at enterprise level. It definitely gives those reliable and cost-effective big data storage and processing platform and an obvious competitive advantage. But along with that there are some risks associated with it. For example, there is a risk of data leakage in transit while it is getting transferred over network from Hadoop client to DataNode.

There are some Hadoop distributors like IBM, Cloudera and Horton works that claims to be providing security to the client's data. Even if their claims are true, not everyone can afford to use a specialized distribution. Getting information security has now become fundamental right in modern era. There should be some open frameworks which anyone canuse easily and get a highly secure Hadoop environment. Also, even if someone is using the services provided by Hadoop distributors, they may have to face some problems. The sensitive data of enterprises is stored on cloud and all the services are accessed through Internet. Since, the data is stored on cloud; companies have to face some security and privacy challenges:

- **Multi-tenancy**

Cloud providers have to build an infrastructure that is cost effective and efficiently scalable to meet customer's requirement. In order to do so they need share storage devices andphysical resources between multiple users. This is called multi-tenancy. But sharing of resources means that attackers can easily target another customer if they both are using the same physical devices and proper security measures are not implemented.

- **Loss of control**

As the companies do not have direct control over their data, they can never know if their data is being used by someone else. This can lead to security issues since there are no transparent mechanisms to monitor the resources directly. There is also a possibility that their data is not completely removed at the time they discontinue using services provided by those service providers.

- **Trust chain in clouds**

As described earlier, customers have to share physical resources with other customers and they do not have direct control over their data. Therefore, customers rely on the cloud providers using trust mechanisms as an alternative to giving users transparent control over their data and cloud resources. Cloud providers needs to build

confidence amongst their customers by assuring them that the provider's operations are certified in compliance with organizational safeguards and standards.

- **Privacy concerns**

Privacy and Security are two distinct domains. Yet they are usually discussed together since security is required in order to provide privacy. The enterprises needs to be sure that their sensitive data is not being accessed by cloud providers and that it is not being share with some third party in return for some money.

There is one more reason for why an organization should secure its Hadoop environment. Security and privacy standards such as International Organization for Standardization (ISO) have evolved. This requires service providers to comply with these regulatory standards to fully safeguard their client's data assets. This has resulted in very protective data security enforcement within enterprises including service providers as well as the clients.

We have discussed some of the major issues that are associated with the use of Hadoop as the big data storage and processing system in this section and why an enterprise needs to secure its Hadoop environment. One of the solutions to overcome those challenges is to encrypt the data. So even if encrypted data in motion is captured by some unauthorized person or it is stored at some remote servers, it will still be safe since unauthorized party will not be able to interpret the actual meaning of the data. Some of the present solutions to achieve security in Hadoop environment and their limitations are discussed in next section.

3. AVAILABLE SOLUTIONS

There are some security measures available for securing sensitive data sets in Hadoop, while they are residing in Hadoop or while they are transferred across the network. In any distributed system, when two parties (the client and server) have to communicate over the network, the first step in this communication is to establish trust between these parties. This is usually done through the authentication process, where the client presents its password to the server and the server varies this password. If the client sends passwords over an unsecured network, there is a risk of passwords getting compromised as they travel through the network. Kerberos is a secured network authentication protocol that provides strong authentication for client/server applications without transferring the password over the network. Kerberos works by using time-sensitive tickets that are generated using the symmetric key cryptography. Kerberos is derived from the Greek mythology where Kerberos was the three-headed dog that guarded the gates of Hades.

The three heads of Kerberos in the security paradigm are:

1) The user who is trying to authenticate.

2) The service to which the client is trying to authenticate.

3) Kerberos security server known as Key Distribution Center (KDC), which is trusted by both the user and the service. The KDC stores the secret keys (passwords) for the users and services that would like to communicate with each other.

- **Encryption for data in motion**

To protect the data in motion, it is required to understand the underlying protocol that is used when data is transferred over the network in Hadoop. A Hadoop client connects to NameNode using the Hadoop RPC protocol over TCP, while the Hadoop client transfers the data to DataNode using the HTTP protocol over TCP. User authentication to Name Node and JobTracker services is through Hadoop's remote procedure call using the SASL framework. Kerberos as discussed previously is used as the authentication protocol to authenticate the users within SASL. The SASL authentication framework can be utilized to encrypt the data while it is being transported into the Hadoop environment thereby protecting the data in motion. SASL security guarantees that data exchanged between the client and servers is encrypted and is not readable by a ``man in the middle''. SASL encryption can be enabled by configuring the property hadoop rpc protection to privacy in core-site.xml. This ensures that the communication between the Hadoop client and NameNode is secured and encrypted.

Any Hadoop client requesting for data from HDFS needs to fetch the data blocks directly from DataNode after it fetches the block ID from NameNode. The Block Access Token (BAT) can be used to ensure that only authorized users are able to access the data blocks stored in DataNodes. There is an excellent compilation of various security mechanisms available for securing Hadoop in [8].

- **Encryption for data at rest**

There are a number of choices for implementing encryption at rest with Hadoop. One of them is using the encryption zone. For transparent encryption, Hadoop has introduced a new abstraction to HDFS: the encryption zone [9]. An encryption zone is a special directory whose contents will be encrypted transparently upon write and transparently decrypted upon read. Here, every encryption zone is composed of single encryption zone key which is specified when the

zone is formed. Each file residing within an encryption zone has its own unique data encryption key (DEK). DEKs are never handled directly by HDFS. HDFS only handles an encrypted data encryption key (EDEK). The Client will perform decrypting an EDEK, and then it will make use of the subsequent DEK to read and write data. HDFS DataNodes simply see a stream of encrypted bytes. A new cluster service should be there to manage encryption keys which is known as the Hadoop Key Management Server (KMS).

In the process of HDFS encryption, the KMS performs three basic responsibilities:

1) Providing the access permission to stored encryption zone keys.

2) Generating new encrypted data encryption keys for storage on the Name Node.

3) Decrypting encrypted data encryption keys for use by HDFS clients. As one can easily identify, this process is similar to what we have seen about Kerberos actual key is never transmitted over network for improved security. For more details about encryption zone and how it can be used to encrypt data at rest, refer [10], [11].

In this section we have discussed about how Kerberos can be used to authenticate users securely and how SASL can be used to encrypt data in transit. We have also learned about encryption zone which is implemented currently in Hadoop for storing encrypted data at rest. So far it seems that Hadoop already have what it takes to give its users a completely secure environment. However, there are some limitations associated with them: Most of times it is not sufficient to just store the data but we must also be able to process the data in efficient manner. The Hadoop encryption zone is not suitable for processing. If we want to run Map Reduce task on data stored in encryption zone then we first have to decrypt the complete file and make it available for Map Reduce task in the decrypted form. Because Hadoop usually deals with large volumes of data and encryption/decryption takes time, it is important that the framework used performs the encryption/decryption fast enough that itdoesn't impact performance. Hadoop encryption zone uses cipher suit like Advanced Encryption Standard (AES) which is good encryption standard but it is definitely having higher memory requirement and can degrade performance since Client node is having limited memory and files used are generally of larger size.

4. PROPOSED FRAMEWORK

In this section we will propose a framework which canhelp eliminate limitations discussed in previous section. This framework does not intend to replace the existing solutions but rather it tries to add flexibility in the current process. This can be added as an additional process that can be usedalong with Kerberos and SASL to give all three-dimensional security to any Hadoop cluster. Three-dimensional security is nothing but the term coined by authors in order to show that by using the proposed framework, anyone can get assured about security of their data in following three ways at the same time:

1) Secure user authentication.

2) Encrypted data in transit.

3) Encrypted data at rest.

One of the major benefits of using this framework is that Map Reduce tasks can directly be executed on data stored in HDFS without requiring the complete file to be decrypted before it can be processed and hence it completely eliminates the use of HDFS encryption zone. However, it should be noted that individual data blocks stored in MapReduce are decrypted at the time of Mapping but that process is faster and memory efficient since size of individual data blocks is slow as compared to the original file. It should also be noted that HDFS encryption zone can still be used if data to be stored there will not frequently be required to be accessed for longer period of time.

The Hadoop client who initiates data read-write operation is also responsible for splitting the complete data set into blocks of size as specified in Hadoop configuration file. Currently the default block size is 128 MB which was previously 64 MB. In the proposed system, the Hadoop Client encrypts each individual block before it is stored into HDFS. This ensures that MapReduce can process each block independently, and the decryption logic is applied during the map phase for a MapReduce job. The decryption key should be made available to the MapReduce job to decrypt the file.

This is provided to the MapReduce program through the job configuration. The solution isscalable and efficient. The solution is most effective if used along with Kerberos and SASL for secure RPC communication and encrypting data in motion. This combination provides the Three-Dimensional Security in Hadoop environment. This approach is shown in the following fig. 1.

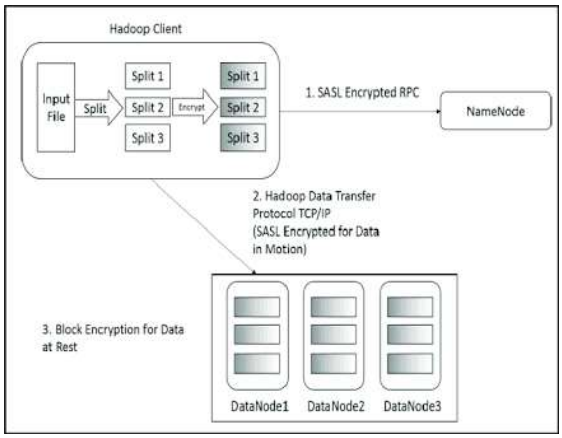

Fig -1: Providing Three-Dimensional Security in Hadoop Environment.

To support block-level encryption in Hadoop, the client should encrypt each individual block before it is transferred to DataNode for storing. MapReduce jobs can directly be executed on encrypted data on HDFS provided that decryption key is known to Mapper class. Similarly, when the file has to be read or copied to local machine, the client will read each individual blocks and then apply the decryption logic in the client side.

An important observation to be made here is that the encryption credentials are not stored in the Hadoop cluster. The encryption credentials are completely managed securely on the client side using some KMS like Kerberos. Now the question arises that what is the algorithm that will be used to encrypt each individual block. This is the most critical part of the process. The algorithms chosen must be memory efficient, faster and secure. Also, Kuber framework is not dependent on a single encryption technique but rather gives client the flexibility to implement their own encryption algorithms based on their needs. Two of the most reliable candidates are AES and Salsa20. The current implementation of the Kuber uses a variant of Salsa20 calledChaCha20 because of the many reasons. Note that ChaCha20 was designed in order improve Salsa20 specifically in terms of diffusion per round and increasing resistance to cryptanalysis, while preserving time per round. Some of the reasonsto choose ChaCha20 instead of AES with initial version of Kuber are:

An efficient implementation of Salsa20 can be as much as 3-5 times faster than AES. ChaCha20 being a variant of Salsa20 gives the same or sometime better speed benefits as Salsa20.

One of the most common attacks in symmetric key:

- Cryptography is differential cryptanalysis. Salsa20 and its variants like ChaCha20 have been proved to be secure against differential cryptanalysis.
- ChaCha20 doesn't require any lookup tables and avoids the possibility of timing attacks
- AES is somewhat complex and troublesome to implement in software. On the other hand, ChaCha20 is easy implement.
- In order to make AES run fast one has to expand the key and pre-compute a series of tables all of these processes increases key setup time and potentially makes vulnerable to cache timing attacks.
- An advantage with block ciphers like AES is that they can randomly access just a portion of encrypted message without decrypting the complete message. Internally, chacaha20 works like a block cipher used in counter mode.
- Consequently it can also be used to randomly access any block of generated key stream. Google's security team has already added the suites to Open SSL.
- The Mozilla team is working on adding it to Firefox. Google also encouraging the increased adaption of the cipher suite in order to offer safer & faster alternatives.
- Salsa20 and its variants like ChaCha20 are free for any use.

5. ALGORITHM DESCRIPTION

ChaCha20 stream cipher is a variant of the Salsa20 family that follows the same design principles as Salsa20 with some changes in the details, most importantly to increase theamount of diffusion per round. The authors also claim that the minimum number of secure rounds for ChaCha20 is smaller than the min imum number of secure rounds forSalsa20. It takes a key (k) of length 32 bytes (256 bits) and a nonce (r) of length 8 bytes (64 bits) as input along with the plaintext. Nonce is a unique value that is never going to change as long as the key is fixed. The pair (k, r) is never used more than once. So we can reuse the key since pair (k, r) is unique. The 256-bit key is then expanded into 264 randomlyaccessible streams, each containing 264 randomly accessible 64-byte blocks. It encrypts a b-byte plaintext by XOR'ing the plaintext with the first b bytes of the stream and discarding the rest of the stream. Decryption operation is same as encryption as a b-byte cipher text is XOR'ed with first b bytes of the stream. This is a common scenario that you will observe with stream ciphers. The quarter round of ChaCha20 is executed as follows.

The algorithm updates four 32-bit state words a, b, c, d as follows:

a C D b; d ^ D a; d <<<D 16;
cC D d; b ^ D c; b <<<D 12;
a C D b; d ^ D a; d <<<D 8;
cC D d; b ^ D c; b <<<D 7;

This code is expressed in common extension of C language. ^means XOR, + means addition modulo 232,

and <<<b means b-bit rotation of a 32-bit integer towards high bits. We can observe here that each word is updated twice in a quarter round of ChaCha20.

As we see, ChaCha20 is very easy to implement, faster on all types of CPUs giving the same level of security as the other standard encryption algorithms such as AES. Also, it should be noted that Kuber framework is not limited to any specific encryption algorithm. Developers have the flexibility to use their own implementation of encryption systems they are comfortable with.

- **Analysis using mahout environment:**

The Apache Mahout offers a library which includes scalable machine-learning algorithms, which are implemented on top of Apache Hadoop by using the MapReduce paradigm. Machine learning is a branch of artificial intelligence which concentrates on permitting machines to learn without being explicitly programmed, and it is commonly used to improve future performance based on previous outcomes.

After big data is kept on the Hadoop Distributed File System (HDFS), Mahout will provide the data science tools in order to automatically find meaningful patterns in those big data sets. The aim of the Apache Mahout project is to make this process faster and easier which will turn big data into big information.

- **What Mahout Does**

Mahout provisions four main data science use cases which are listed below:

Collaborative filtering – It mines user behavior and outputs product recommendations (e.g. Amazon recommendations).

Clustering – It will collect items in a particular class (such as web pages or newspaper articles) and then it will organize them into naturally occurring groups, such that items which belongs to the same group are similar to each other.

Classification – It will learn from existing categorizations and then allocates unclassified items to the best category.

Frequent itemset mining – It analyzes items in a group (e.g. items placed at shopping cart or terms in a query session) and then identifies which items will appear together.

- **Features of Mahout**

  - The algorithms of Mahout are written on top of Hadoop. Hence, it works fine in distributed environment. Mahout utilizes the Apache Hadoop library in order to support scalability effectively in the cloud.
  - Mahout provides the coder a ready-to-use framework for performing data mining tasks over large volumes of data.
  - Mahout allows applications to analyze large sets of data effectively and in less time.
  - It consists of numerous MapReduce facilitated clustering implementations for example k-means, fuzzy k-means, Canopy, Dirichlet, and Mean-Shift.
  - Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
  - It provides distributed fitness function capabilities for evolutionary programming.
  - Includes matrix and vector libraries.

- **Applications of Mahout**

  - The Companies which includes Adobe, LinkedIn, Facebook, Twitter, Foursquare and Yahoo are using Mahout internally.
  - Foursquare-this helps in finding out places, food, and entertainment that are offered in a particular area. It utilizes the recommender engine of Mahout.
  - Twitter makes use of Mahout for user interest modelling.
  - Yahoo! utilizes Mahout for pattern mining.

- **Applying Encryption over data to be clustered:**

In this work the data to be clustered is encrypted before uploading to HDFS. Then the data is uploaded to HDFS in encrypted format.The encrypted data is decrypted before clustering. The clustering is performed after this process and the time taken for the encryption process is analyzed. Thus it ensures that the data to be clustered will be securely stored in Hadoop environment.

6. RESULTS AND DISCUSSION

Having decided to use ChaCha20, We started to experiment with the speed of the algorithm with files of different sizes and different buffersizes which will be given as an input to the algorithm. The results are noted in Table 1.

Table -1: Speed (Mb/s) of encryption with various file sizes' (rows) and buffer sizes' (columns).

	1 KB	100 KB	500 KB	1 MB	64 MB	128 MB
825 KB	26 Mb/s	20 Mb/s	**36 Mb/s**	34 Mb/s	18 Mb/s	16 Mb/s
4.29 MB	32 Mb/s	32 Mb/s	56 Mb/s	**70 Mb/s**	56 Mb/s	40 Mb/s
60.5 MB	76 Mb/s	68 Mb/s	124 Mb/s	**140 Mb/s**	92 Mb/s	78 Mb/s
381 MB	80 Mb/s	78 Mb/s	150 Mb/s	**152 Mb/s**	56 Mb/s	56 Mb/s
1.00 GB	64 Mb/s	60 Mb/s	**142 Mb/s**	84 Mb/s	60 Mb/s	58 Mb/s
2.49 GB	54 Mb/s	78 Mb/s	78 Mb/s	**80 Mb/s**	56 Mb/s	56 Mb/s

Configuration: Intel Core i3 CPU with 4 GB RAM running on Windows 8.1.

The experiment started with an assumption that increasing the buffer size will reduce the iterations of loop and consequently reduce the overall encryption time. However, the results were somewhat different. The highest speed was achieved with buffer size between 500 KB and 1 MB. It is because the cost of machine cycles associated with transferring larger blocks like 64 MB from one method to other isgreater than the time saved by reducing the number of iterations of the loop. Fig.2. shows the graphical representation of the results. The highest speed in each case is achieved when buffer size of 500 KB or 1 MB is used.

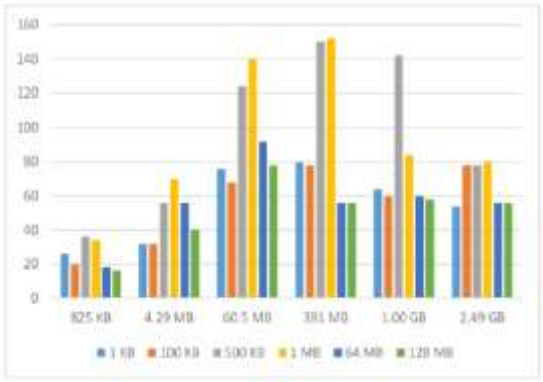

Chart -1: Encryption speed in Mb/s of 6 files with various buffer sizes.

ChaCha20 is designed such that it can encrypt any random block of input data, making it suitable to be used as a block cipher. Another major advantage of this method is that each encrypted block can be individually decrypted provided you have the key and nonce for the decryption. Consequently, MapReduce tasks can be executed directly on encrypted data stored in HDFS. The complete process for large scale encryption in Hadoop environment can be summarized as follows:

The Hadoop client configures the key and nonce for encryption process in KMS. Data to be stored in HDFS is first encrypted at client side by dividing it into smaller chunks using parallel computing.

MapReduce tasks can be directly executed on encrypted data. The level of flexibility can be imagined by the fact that any current implementation of MapReduce can be adapted to use Kuber with only two lines of modification in their original program. All they have to do is to import corresponding decryption library and call decryption method before processing the data. Authorized clients can access the encrypted file stored in HDFS. Decryption process is similar to encryption process.

7. CONCLUSION

Data is growing at very high speed in the present world. One of the finest and most popular technologies available for handling and processing that enormous amount of data is the Hadoop ecosystem. Enterprises are increasingly relying on Hadoop for storing their valuable data and processing it. However, big data exposes the enterprise to numerous data security threats. In this work encryption and decryption technology is applied over data before uploading to the HDFS, thereby providing security. The Chacha20 algorithm is used for this process. The analysis is done using mahout environment over the data to be clustered. The results are observed. Thus this work presents an effective technique for securing the big data to be clustered in hadoop Environment.

REFERENCES

1. S. Ghemawat, H. Gobioff, and S.-T. Leung, ``The Google _le system,'' in Proc. 19th ACM Symp. Oper. Syst. Principles (SOSP), 2003, pp. 29_43.

2. S. Ghemawat and J. Dean, ``MapReduce: Simpli_ed data processing on large clusters,'' ACM Commun. Mag., vol. 51, no. 1, pp. 107_113, Jan. 2008.

3. D. Borthakur, ``The Hadoop distributed file system: Architecture anddesign,'' Hadoop Project Website, vol. 11, p. 21, Aug. 2007.

4. T. White, Hadoop: The De_nitive Guide. Farnham, U.K.:O'Reilly, 2012.

5. D. de Roos, P. C. Zikopoulos, R. B. Melnyk, B. Brown, and R. Coss, Hadoop For Dummies. Hoboken, NJ, USA: Wiley, 2014.

6. B. Lakhe, Practical Hadoop Security. New York, NY, USA: Apress, 2014, pp. 1946.

7. Apache Hadoop, accessed Dec. 2016. [Online]. Available: https://en.wikipedia.org/wiki/Apache Hadoop History.

8. P. P. Sharma and C. P. Navdeti, ``Securing big data Hadoop: A review of security issues, threats and solution,'' Int. J. Comput. Sci. Inf. Technol.,vol. 5, no. 2, pp. 2126_2131, 2014.

9. Apache Hadoop 2.7.3_Transparent Encryption in HDFS, accessed Feb. 2017. [Online]. Available: https://hadoop.apache.org/docs/r2.7.2/ hadoop-project-dist/hadoop-dfs/TransparentEncryption.html.

10. HDFS Data At Rest Encryption, accessed Feb. 2017. [Online].Available:https://www.cloudera.com/documentation/enterprise/5x/topics/cdh_sg_hdfs_encryption.html.

11. HDFS Encryption Overview, accessed Feb. 2017. [Online].Available:https://docs.hortonworks.com/HDPDocuments/HDP2/HDP2.3.2/bk_hdfs_admin_tools/content/hdfs-encryption-overview.html.