Securing Hadoop using OAuth 2.0 and Real Time Encryption Algorithm

Mr. Vijaykumar N. Patil
Department of Computer Engineering, Savitribai Phule Pune University
SKN Sinhgad Institute of Technology & Science, Lonavala, Pune, Maharashtra, India,
vijay.patil.karad@gmail.com

Prof. Venkatesan N.
Department of Computer Engineering, Savitribai Phule Pune University
SKN Sinhgad Institute of Technology & Science, Lonavala, Pune, Maharashtra, India,
venktesann.sknsits@sinhgad.edu

***Abstract*—**Hadoop is most popularly used distributed programming framework for processing large amount of data with Hadoop distributed file system (HDFS), but processing personal or sensitive data on distributed environment demands secure computing. Originally Hadoop was designed without any security model. Hadoop projects deals with security of data as a top agenda, which in turn to represents classification of a critical data item. The data from various applications such as financial deemed to be sensitive which need to be secured. With the growing acceptance of Hadoop, there is an increasing trend to incorporate more and more enterprise security features. The encryption and decryption technique is used before writing or reading data from HDFS respectively. Advanced Encryption Standard (AES) enables protection of data at each cluster which performs encryption or decryption before read or writes occurs at HDFS. The earlier methods do not provide Data privacy due to the similar mechanism used to provide data security to all users at HDFS and also it increases the file size; so these are not suitable for real-time application. Hadoop require additional terminology to provide unique data security to all users and encrypt data with the compatible speed. We have implemented method in which OAuth does the authentication and provide unique authorization token for each user which is used in encryption technique that provide data privacy for all users of Hadoop. The Real Time encryption algorithms used for securing data in HDFS uses the key that is generated by using authorization token.

***Keywords*-** *Hadoop,DataNode,NameNode,TaskTracker,ASE,HDFS,OAuth.*

*****

I. INTRODUCTION

Hadoop was developed from GFS (Google File System) [2, 3] and MapReduce papers published by Google in 2003 and 2004 respectively. It has been popular recently due to its highly scalable distributed programming or computing framework, it enables processing big data for data-intensive applications as well as many analytics. Hadoop is a framework of tools which supports running application on big data and it is implemented in java. It provide MapReduce programming architecture with a Hadoop distributed file system(HDFS), which has massive data processing capability with thousands of commodity hardware's by using simply its map and reduce functions. Since Hadoop is usually executing in large cluster or may be in a public cloud service. Like Yahoo, Amazon, Google, etc. are such public cloud where many users can run their jobs using Elastic MapReduce and cloud storage that is used as Hadoop distributed file system, it is essential to implement the security of user data on such storage or cluster. Hadoop project during its early design stage the simple security mechanisms are employed such as file permissions and access control list [4].Encryption and decryption is key means for securing Hadoop file system(HDFS), where many DataNodes (or clusters that is originally DataNodes) store file to HDFS, those are transferred while executing MapReduce (user submitted program) job. It is reported that upcoming Hadoop software or version will include encryption and decryption functionality [5]. In today's era, internet now initiate huge amount of data every day. The IDC's publish a statistics analysis in 2012 it include the structured data on the internet is about 32% and unstructured is 63%. Also the volume of digital content on internet grows up to more than 2.7 ZB in 2012 which is up 48% from 2011 and now rocketing towards more than 10 ZB by 2015. Every industry and business organizations are now an important data about different product, production and its market survey which is a big data beneficial for productivity growth. In commercial data analysis application which is operate on big data the Hadoop becomes defacto platform, in upcoming 5 year, more than 70% of big data applications are running on Hadoop.

The overall system architecture is shown in Figure1.The files on Hadoop file system (HDFS) are split into different blocks and replicated with multiple DataNodes to ensure high data availability and durability to failure of execution of parallel application in Hadoop environment. Originally Hadoop clusters have two types of node operating as master-salve or master-worker pattern [6]. NameNode as a master and DataNodes are workers nodes of HDFS. Where data files are actually located in Hadoop is known as DataNode which only leads storage. However NameNode contains information about where the different file blocks are located but it is not persistent, when system starts block may changes one DataNode to another DataNode but it report to NameNode or client who submit the MapReduce job or owner of Data periodically [11]. The communication is in between DataNode and client NameNode only contains metadata.

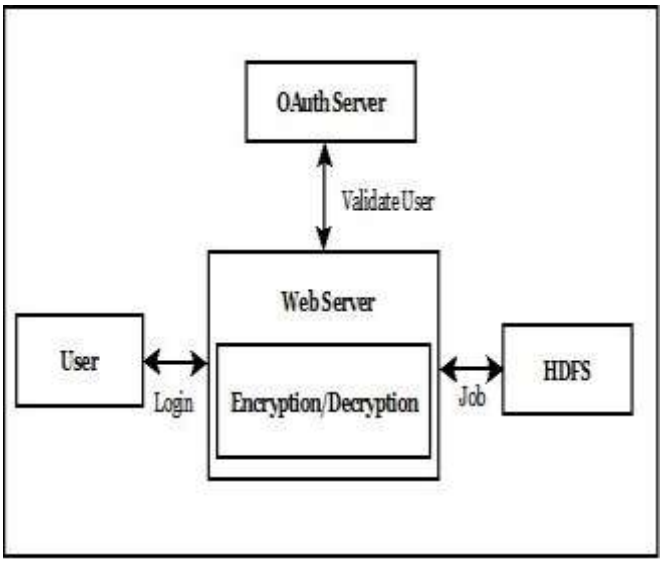

Figure 1: Overall System Architecture

The organization of this paper is as follows: Chapter 2 focuses on related work done in Hadoop security with its pros and cons. Chapter 3 gives detailed description of proposed work with implementation. Chapter 4 gives experimental setup and results of proposed system. At last Chapter 5 concludes the paper and focuses on future direction to our work.

II. RELATED WORK

Hadoop is originally a distributed system which allows us to store big data and supports for processing it in parallel environment. Many organizations uses big data applications to predict future scope, Hadoop cluster store the sensitive information about such organizations (information like productivity, financial data, customer feedback etc.). As result Hadoop cluster require strong authentication and authorization with data protection such as encryption

The method proposed in [1] is a secure Hadoop architecture in which encryption and decryption functions are added to the HDFS. Also HDFS is secured by adding the AES encrypt/decrypt class in Hadoop.

The trusted computing technologies [2] combined with the Apache Hadoop Distributed File System (HDFS) in an effort to address concerns of data confidentiality and integrity. The two different types of integrations called HDFS-RSA and HDFS-Pairing [3] used as extensions of HDFS, these integrations provide alternatives toward achieving data confidentiality for Hadoop.

Novel method used [4] to encrypt file while being uploaded. Data read from file is transferred to HDFS across a buffer. In this approach, an encryption, which is transparent to user, is applied to the buffer's data before being sent to an out stream to write to HDFS. Thus, user needs not to worry about the data's confidentiality anymore.

The homomorphic encryption technology [5] enables the encrypted data to be operable to protect the security of the data and the efficiency of the application. The authentication agent technology offers a variety of access control rules, which are a combination of access control mechanisms, privilege separation and security audit mechanisms, to ensure the safety for the data stored in the Hadoop file system

These above mentioned techniques provide good security to HDFS but Hadoop is a distributed programming framework for processing large data where the DataNodes are physically distributed with its individual tasks and also the task given by TaskTracker, demands for more secure computing. All above described methods does not provide Data privacy due to the similar mechanism used to provide data security to all users at HDFS. The size of encrypted data after using AES or similar algorithm is greater, so these are not efficient where file storage grows quickly because of performance overhead. If we use the encryption technique which provide data privacy and also does not affect size of data too much so it support for real time application and possible to reduce overhead occurs in existing system.

III. PROPOSED SYSTEM

We have proposed new method to secure data at HDFS by analyzing all older methods described above. It is implemented by using OAuth (called Open Standard for Authorization) and Real Time Encryption Algorithm. OAuth 2.0 is an Open Authentication Protocol that helps to run-over the problems of conventional client-server authentication model. In the conventional client-server model, the client requests to an access protected resource on the server by authenticating itself using the resource owner's passport. In order to give third-party applications access to restricted resources, the resource owner verifies its authorization with the third-party [13].

In proposed system OAuth 2.0 is used to authenticate user as well as it return unique token for each user who attempt successful login. The token returned by OAuth server used in encryption method so it provides data confidentiality and integrity to the user data. The files are encrypted before load to HDFS and decrypted when job execution is in progress [1]. The Real Time Encryption Algorithm use the OAuth token as key and Encrypt data by XoRing with the key.

Detailed System Architecture shown in Figure 2, the User login in to system then gives 'n' number of documents as a input to the HDFS but before write to HDFS will send that data to Real Time encryption model which will process the data and perform data encryption, similarly it also perform decryption when MapReduce job read data from HDFS at time of execution of job. OAuth provide authentication token and authorization token which are used for user verification and encryption/decryption algorithm respectively.

ISSN: 2321-8169
4382 - 4387

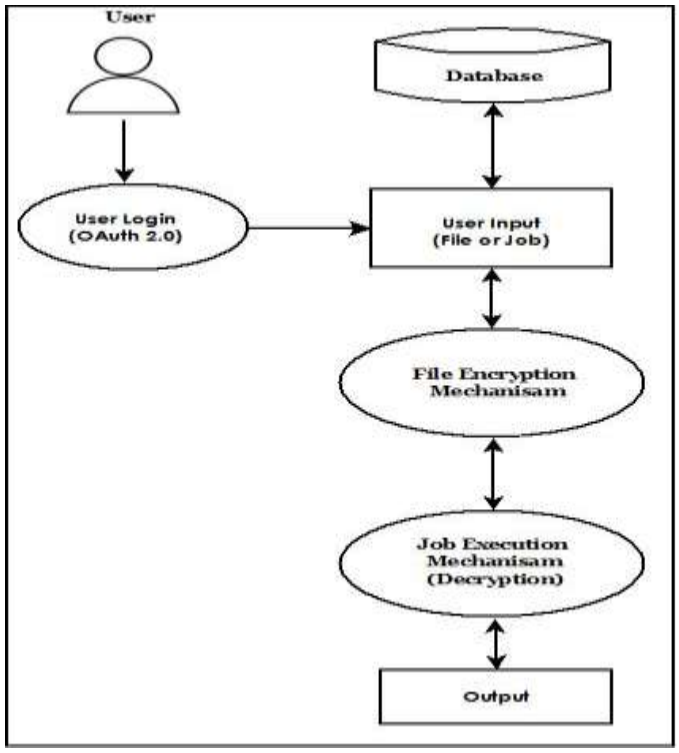

Figure 2: Detailed System Architecture

A. *Algorithms for OAuth Authorization Server*

Input: User Credentials

Output: Authentication token & authorization token

The following steps explain the server-side flow:

1. *Start*
2. *Obtain an access token.*
3. *User decides whether to grant access to your application*
4. *OAuth Server redirects user to your application*
5. *Exchange authorization code for refresh and access tokens.*
6. *Process response and store tokens*
7. *Stop*

The following steps explain the client-side flow:

1. *Start*
2. *Obtain an access token.*
3. *Server decides whether to grant access to your application*
4. *OAuth Server redirects user to your application*
5. *Validate the user's token*
6. *Process the token validation response.*

B. *Real Time Encryption Algorithm*

Encryption Steps

1. *Start*
2. *Retrieve OAuth token at successful user login*
3. *Generate random key using key generator*
4. *Read data from file and XoRing with the key*
5. *Add key in the XoRed data, which generated by key generator*
6. *Write encrypted data in file and load file to HDFS*
7. *Stop*

Decryption Steps

1. *Start*
2. *Retrieve OAuth token at successful user login*
3. *Generate random key using key generator*
4. *Read data from file and XoRing with the key*
5. *Add key in the XoRed data, which generated by key generator*
6. *Write encrypted data in file and load file to HDFS*
7. *Stop*

C. *Mathematical Model using Set Theory*

1. *Let S= {} be as a secure Hadoop system*
2. *Obtain an OAuth authentication tokens AT*

   AT = {uid_at1, uid_at2....., uid_atn}

   Where uid_at1= unique token for specific user.

   S= {AT}
3. *Obtain an OAuth authorization tokens OT*

   OT = {uid_ot1, uid_ot2,.....,uid_otn}

   Where uid ot1= unique token for specific user.

   S= {AT, OT}

ISSN: 2321-8169
4382 - 4387

4. *Give input files upload to HDFS F*

   F= {f1, f2 ...fn}

   Where f1is a text file

   S= {AT, OT, F}

5. *Perform encryption process on set of files is a En*

   En= {F, OT}

   Where En process take input as set of files and user authorization token

   S = {AT, OT, F, En}

6. *Perform decryption process on set of files is a Dn*

   Dn = {F, OT}

   Where Dn process take input as set of files and user authorization token

   S = {AT, OT, F, En, Dn }

7. *Identify MapReduce job to analyze data at HDFS*

   J = {j1_dn,j2_dn,.....,jn_dn}

   Where j1_dn is a MapReduce program with decryption process

   S = {AT, OT, F, En, Dn, J}

8. *Final Set S = {AT, OT, F, En, Dn, J}*

D. *Mathematical Model for proposed system*

1. *Initialize Tokens*

   A) At = {}

   B) Ot = {}

2. *Initialize path/files upload to HDFS*

   F = {}

3. *Process encryption module*

   En= fp, uid_otn

   Where $fp \in F$

   $uid\_otn \in Ot$

4. *Execute job J = Fc, uid_otn*

   Where $Fc \in En$

5. *Encrypted files obtained by equation*

$$S(En) = \sum_{n+1}^{fn} \text{fp^uid\_ot}$$

   Where n is total number of files in a file set F={},

   fp is the plain text file and uid_ot is a user

   Authorization token

6. *Job execution obtained by equation*

$$S(Dn) = \sum_{n+1}^{fn} \text{fc^uid\_ot}$$

   Where n is total number of files in a file set F= {}

   fc is the cipher text file and uid_ot is a user

   Authorization token

IV. Experimental Setup and Results

To carry out the experiment we have installed Ubuntu Linux 12.04. Openjdk1.7 and Apache Tomcat 1.7 installed in it and SSH enabled. Hadoop 1.2.1 have been configured as a Single-Node Cluster to use the HDFS and MapReduce capabilities. To setup OAuth server we deploy and configure OAuth app [17] for login with Google and also deploy another app [18] for login with Facebook. The NameNode structure is given in figure 3.

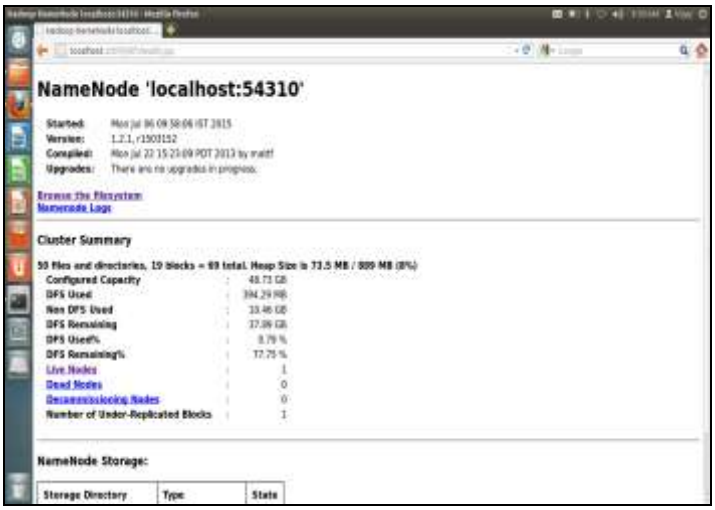

Figure 3: NameNode Structure

ISSN: 2321-8169
4382 - 4387

The NameNode is center piece of Hadoop in light of the fact that it controls the entire DataNodes exhibit in bunch. It is a Single-Point-of-Failure yet late forms (0.21+) accompany Backup NameNode [2] to make it exceptionally accessible. The DataNodes contain all the information in bunch on which we will work our MapReduce projects and perspective the activity information from different points of view. JobTracker controls all the tasks which are running on TaskTrackers shown in following figure 4.

Figure 4: TaskTracker

We have developed two different encryption techniques first does encryption using AES and second new algorithm perform encryption using OAuth token we called as Real-time encryption algorithm. The MapReduce programs (Hadoop job) which take the input as encrypted data and execute job, we can observe that 23.0490 seconds was taken for running a WordCount MapReduce job for unencrypted HDFS for size of 10MB test file while 83.2780 seconds for the encrypted HDFS with AES and 54.2360 seconds for encrypted HDFS with Real-time encryption algorithm(RTEA).

Data (MB)	Encryption Type	Encrypted Data(MB)	Time Consume for Encryption (sec)	Time Consume to upload to HDFS(sec)
1	AES	1.8819	26.2190	1.7660
	RTEA	1.0659	12.1510	1.6370
10	AES	20.1015	298.0950	2.0110
	RTEA	10.7252	131.5510	1.8120

Table 1: Comparison between AES and Real Time Algorithm

Table 1 shows the file encryption Comparison between AES and the new Algorithm. The result of data uploads of plain file and encrypted file shown in following figures in terms of graphs. The job execution Comparison between AES encryption and the new Algorithm shows in Table 2. The results are shown in following figures in terms of graphs.

Data (MB)	Encryption Type	Encrypted Data(MB)	Time Consume for job Execution(sec)
1	AES	1.8819	26.0420
	RTEA	1.0659	22.0510
10	AES	20.1015	83.2780
	RTEA	10.7252	54.2360

Table 2: Comparison between job executions of AES encrypted data and Real Time Encryption Algorithm

Figure 5: Shows graph of time required to encrypt input file using AES and Real Time encryption algorithm

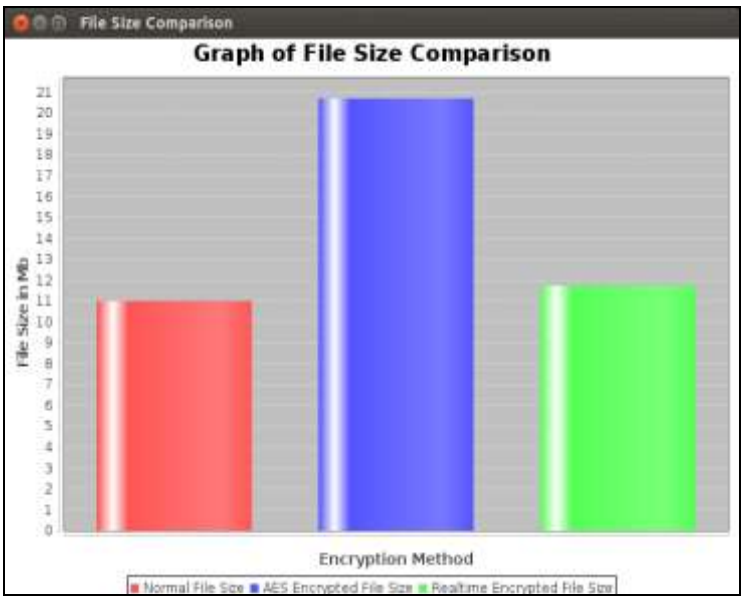

Figure 5: Shows graph of comparison of original file size and file size encryption using AES and Real Time encryption algorithm

 ISSN: 2321-8169
4382 - 4387

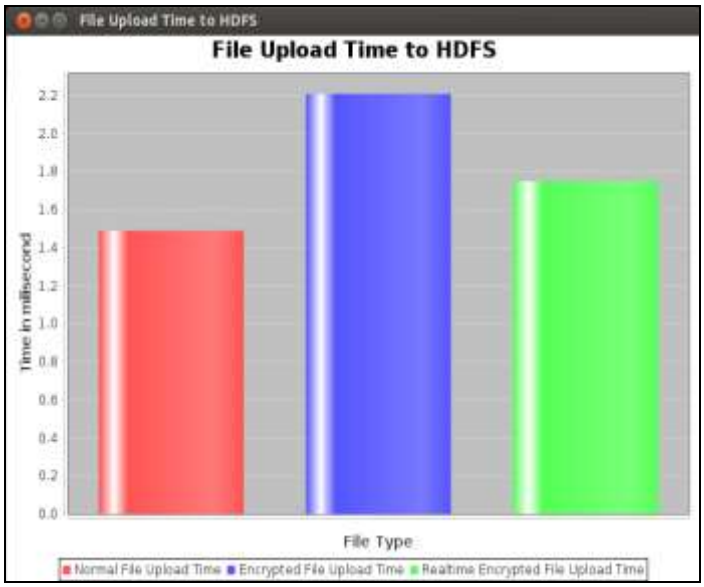

Figure 6: Shows graph of comparison of file upload time of original file and files after encryption using AES and Real Time encryption algorithm

Figure 7: Shows graph of comparison of job execution time of original file and files after encryption using AES and Real Time encryption algorithm

V. CONCLUSION AND FUTURE WORK

In the today's world of Big Data, where data is gathered from different sources in such case, the security is a measure issue, as there does not any fixed source of data and HDFS not have any kind of security mechanism. Hadoop adopted by various industries to process such big amount and sensitive data, demands strong security mechanism.

Thus encryption/decryption, authentication, authorization are the methods those much helpful to secure Hadoop file system.

In Future work our topic leads to generate Hadoop with all kinds of security mechanism for securing data as well as secure job execution.

VI. ACKNOWLEDGMENT

We would like to thank IJRITCC for giving such wonderful platform for the PG students to publish their research work. Also would like to thanks to our guide & respected teachers for their constant support and motivation for us. Our sincere thanks to SKN Sinhgad Institute of Technology and Science for providing a strong platform to develop our skill and capabilitie.

VII. REFERENCES

[1] Seonyoung Park and Youngseok Lee, Secure Hadoopwith Encrypted HDFS, Springer-Verlag Berlin Heidelberg in 2013

[2] [2] Dean J., Ghemawat S.: MapReduce: Simplified DataProcessing on Large Cluster, In:OSDI (2004)

[3] Ghemawat S., Gobioff H., Leung, S.: The Google FileSystem. In: ACM Symposium onOperating SystemsPrinciples (October 2003)

[4] OMalley O., Zhang K., Radia S., Marti R., Harrell C.: Hadoop Security Design,Technical Report (October2009)

[5] White T.: Hadoop: The Definitive Guide, 1st edn.OReilly Media (2009)

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

[7] Jason Cohen and Dr. Subatra Acharya Towards aTrusted Hadoop Storage Platform:Design Considerationsof an AES Based Encryption Scheme with TPM RootedKeyProtections. IEEE 10th International Conference on Ubiquitous Intelligence & Computing in 2013

[8] Lin H., Seh S., Tzeng W., Lin B.P. Toward DataConfidentiality via Integrating Hybrid EncryptionSchemes and Hadoop Distributed FileSystem. 26th IEEE International Conference on Advanced Information Networking and Applications in 2012

[9] Thanh Cuong Nguyen, Wenfeng Shen, Jiwei Jiang andWeimin Xu A Novel Data Encryption in HDFS. IEEE International Conference on Green Computing and Communicationsin 2013.

[10] Devaraj Das, Owen OeMalley, Sanjay Radia and KanZhang Adding Security to Apache Hadoop. in hortanworks

[11] Songchang Jin, Shuqiang Yang, Xiang Zhu, and Hong Yin Design of a Trusted File System Based on Hadoop .Springer-Verlag Berlin Heidelberg in 2013

[12] [AdvancedEncryption Standard,http://en.wikipedia.org/wiki/Advanced Encryption Standard

[13] Sharma Y. ; Kumar S. and Pai R.M; FormalVerification of OAuth 2.0 Using Alloy Framework .International Conference on Communication Systems and Network Technologies in 2011

[14] Ke Liu and Beijing Univ OAuth BasedAuthentication and Authorization in Open Telco API .IEEE International Conference on Communication Systems and Network Technologies in 2012

[15] Big Data Security: The Evolution of Hadoops Security ModelPosted by Kevin T. Smith on Aug 14, 2013

[16] Securing Big Data Hadoop: A Review of Security Issues, Threats and Solution by Priya P. Sharma and Chandrakant P. Navdeti in 2014

[17] https://console.developers.google.com

[18] https://developers.facebook.com/apps