International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 04 | Apr-2016 www.irjet.net p-ISSN: 2395-0072

Data Security in Hadoop Distributed File System

Sharifnawaj Y. Inamdar[1], Ajit H. Jadhav[2], Rohit B. Desai[3],

Pravin S. Shinde[4], Indrajeet M. Ghadage[5], Amit A. Gaikwad[6].

[1]*Professor,Department of Computer Science & Engineering, DACOE Karad, Maharashtra, India*
[2,3,4,5,6]*Student,Final Year B.E.-Computer Science & Engineering, DACOE Karad, Maharashtra, India*

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.

In this project, security of HDFS is implemented using encryption of file which is to be stored at HDFS. For encryption a real-time encryption algorithm is used. So a user who has the key for decryption can perform decryption of data & access that data for data mining. User authentication is also done for the system. We have also compared this method with the method previously implemented i.e. encryption & decryption using AES. Encrypting using AES results into growing of file size to double of original file & hence file upload time also increases. The technique used in this project removes this drawback.

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.

Key Words: Hadoop,Big data,Security,HDFS,OAuth.

1.INTRODUCTION

Hadoop was developed from GFS (Google File System) [2, 3] and Map Reduce papers published by Google in 2003 and 2004 respectively. Hadoop is a framework of tools, implemented in Java. It supports running applications on big data.

1.1 Project Idea:

Hadoop is designed without considering security of data. Data stored at HDFS is in plaintext. This data is prone to be accessed by unauthorized user. So method for securing this data is needed. Hence we are developing this highly secure system for Hadoop Distributed File System.

1.2 Need of project:

Hadoop is generally executing in big clusters or might be in an open cloud administration. Amazon, Yahoo, Google, and so on are such open cloud where numerous clients can run their jobs utilizing Elastic MapReduce and distributed storage provided by Hadoop. It is key to execute the security of client information in such systems.

Web produces expansive measure of information consistently. It incorporate the organized information rate on web is around 32% and unstructured information is 63%. Additionally the volume of advanced substance on web grows up to more than 2.7ZB in 2012 which is 48% more from 2011 and now soaring towards more than 8ZB by 2015. Each industry and business associations are has a critical information about various item, generation and its business sector review which is a major information advantageous for efficiency development.

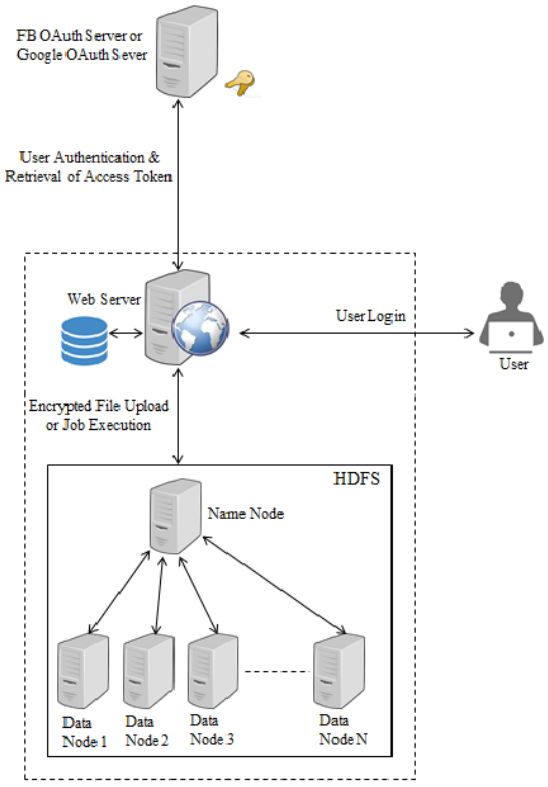

Fig-1:System Architecture

The files in Hadoop distributed file system (HDFS) are divided into multiple blocks and replicated to other DataNodes(by default 2 nodes) to ensure high data availability and durability in case of failure of execution of job(parallel application in Hadoop environment). Originally Hadoop clusters have two types of node operating as master-

salve or master-worker pattern [6]. NameNode is a master node and DataNodes are workers nodes in HDFS. Data nodes are the nodes where actual file(part of file on a node) is stored. 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]. Client gets list of data nodes where file blocks reside & then communicate with Data nodes only. NameNode contains only metadata. Our proposed system architecture is as shown in fig-1.

2. RELATED WORK

Hadoop is a distributed system which permits us to store enormous structured & unstructured information(i.e. Big Data). It is also helpful to process such huge amount of data in parallel environment. Numerous associations utilizes huge information applications to foresee future degree, Hadoop group store the sensitive data about such associations (data like profitability, monetary information, client criticism and so forth.). As result Hadoop file system requires method to protect such information using very strong authentication. It also requires authorization of user.

The technique described in [1] is a secure Hadoop architecture where encryption and decryption functions are applied to the HDFS. AES encrypt/decrypt classes are added for encryption and decryption of data.

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. In this method, data which is to be uploaded to HDFS is first stored in a buffer. After that encryption is applied to the buffer's data before being sending it to HDFS. This encryption is transparent to user. Thus, client needs not to stress over the information's privacy any longer.

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 provides various access control rules, which are defined using access control mechanisms, privilege separation and security audit mechanisms, to ensure the protection for the data that will be stored in the HDFS.

These aforementioned systems give good security to HDFS however Hadoop is a distributed programming framework for processing huge information where the DataNodes are physically appropriated with its individual tasks furthermore the undertaking given by TaskTracker, requests for more secure processing of data. All above portrayed techniques does not give Data protection because of the comparative instrument used to give information security to all clients at HDFS. The measure of scrambled information in the wake of utilizing AES or comparative algorithm is more noteworthy, so these are not proficient where record stockpiling becomes rapidly on account of execution overhead. In the event that we utilize the encryption procedure which give information protection furthermore does not influence size of information an excessive amount of so it support for ongoing application and conceivable to diminish overhead happens in existing framework.

3. PROPOSED SYSTEM

We have proposed new technique for securing data at HDFS by analyzing all techniques previously mentioned. It is actualized by utilizing Real Time Encryption Algorithm and OAuth (called Open Standard for Authorization). OAuth 2.0 is an Open Authentication Protocol that is used for authentication and authorization of client in conventional client-server model. In the traditional client-server model, the customer solicitations to an entrance secured asset on the server by verifying itself utilizing the asset proprietor's international ID. 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, to authenticate user we have used OAuth 2.0, which returns unique token for each user who attempts successful login. The token returned by OAuth server utilized as a part of encryption strategy so it gives information privacy 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 utilizes the OAuth token as key and Encrypt data (uploaded by user) by XoRing with the key.

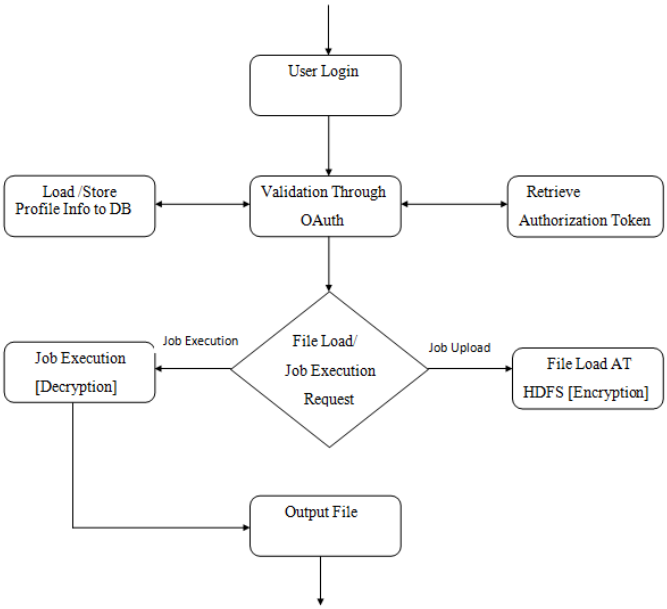

Fig-2:Flow chart

Flow chart is shown in Fig-2. User does log in to system using OAuth 2.0 and then uploads 'n' number of

documents(either file or job) as an input to the HDFS. But before writing to HDFS it will be passed to Real Time encryption model. In this model data will be encrypted. Similarly decryption will be performed when MapReduce job read data from HDFS after job execution request. Authentication token and authorization token provided by OAuth are used for user verification and encryption/decryption algorithms respectively.

A. ***Algorithms in OAuth Protocol***
Input: Login ID & Password(third party) of client
Output: Authorization token & Authentication token
The following steps are executed at the server-side:
1. Begin

2. Get an access token.

3. Client chooses whether to give access to your application

4. Client is redirected to your application by OAuth Server

5. Exchange authorization code for refresh and access tokens.

6. Process response and store tokens

7. Stop

The following steps are executed at client-side:
1. Begin

2. Get an access token

3. Server verifies credentials & grant access to your application

4. Client is redirected to your application by OAuth Server

5. Validation of the client's token

6. Token validation response is processed.

7. Stop

B. ***Real Time Encryption Algorithm***

Encryption algorithm

1. Begin

2. Retrieve OAuth token after successful user login

3. Generate key using random key generator

4. Read data from file and XoR that data with the key, which generated by key generator

5. Append the key to the XoRed data

6. Write encrypted data in a file and load that file to HDFS

7. Stop

Decryption algorithm

1. Begin

2. Retrieve data for decryption

3. Extract key from data

4. Read remaining data from file and XoR it with the key

5. Pass decrypted data to MapReduce job submitted by client

6. Combine the output from all working nodes & send it to user

7. Stop

4. TEST SETUP AND RESULTS

To do the experiment we have installed Ubuntu Linux 12.04 on our machine. After that we installed Openjdk1.7 and Apache Tomcat 1.7 and enabled SSH. We configured Hadoop 1.2.1 as a Single-Node Cluster to use the HDFS and MapReduce capabilities. For OAuth server setup we deployed and configured OAuth app [17] for login with Google and also deployed another app [18] for login with Facebook.

The NameNode is focus bit of Hadoop in light of the way that it controls the whole DataNodes exhibit in a cluster. It is a Single-Point-of-Failure yet late structures (0.21+) go with Backup NameNode [2] to make it outstandingly available. The DataNodes in HDFS contain all the data on which we be input to our MapReduce jobs. JobTracker at NameNode controls all the tasks which are running on TaskTrackers.

We have implemented two different encryption techniques of which first does the encryption using AES and second algorithm perform the encryption using OAuth token. We named the second algorithm as Real-time encryption algorithm. The MapReduce programs (Hadoop job) which take the encrypted data as input and execute job, we observed that it took 23.0490 seconds to execute a WordCount MapReduce job for the unencrypted HDFS(normal execution) for size of 10MB test file, while it took 83.2780 seconds for the encrypted HDFS using AES and 54.2360 seconds taken for encrypted HDFS using Real-time encryption algorithm(RTEA).

Table -1: Comparison between AES & RTEA for encryption

Data (MB)	Encryption Type	Encrypted Data(MB)	Time taken for Encryption(sec)	Time taken 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 shows the Comparison between AES and Real-time encryption Algorithm for file encryption. The results of data uploads of plain file and encrypted file is shown in graphs. The job execution time Comparison between AES encryption and the Real-time encryption Algorithm is shown in Table 2. The results of the tests are shown in graphs(figures 3-6).

Table -2: Comparison between AES & RTEA for job execution

Data (MB)	Encryption Type	Encrypted Data(MB)	Time taken to execute job(sec)
1	AES	1.8819	26.0420
	RTEA	1.0659	22.0510
10	AES	20.1015	83.2780
	RTEA	10.7252	54.2360

Fig-3: Graph of encryption time taken for input file using AES & RTEA

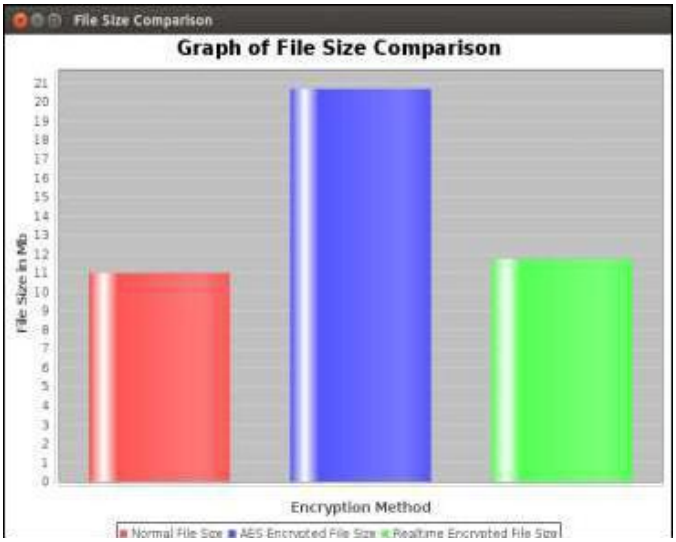

Fig-4: Graph of file size comparison

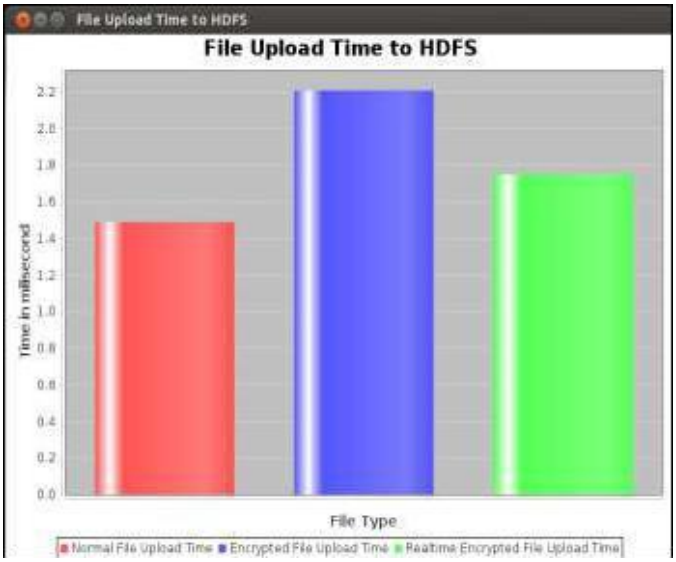

Fig-5: Graph of comparison of file upload time to HDFS

Fig-6: Graph of comparison of job execution time

5. CONCLUSIONS

In the today's world of Big Data, where information is assembled from various sources in such case, the security is a noteworthy issue, as there does not any altered wellspring of information and HDFS not have any sort of security system. Hadoop embraced by different commercial enterprises to process such enormous and delicate information, requests solid security system.

Along these lines encryption/decryption, authentication & authorization are the techniques those much supportive to secure information at Hadoop Distributed File System.

In Future work our subject prompts produce Hadoop with a wide range of security techniques for securing information and additionally secure execution of job.

ACKNOWLEDGEMENT

We would like to thank IRJET for giving such wonderful platform for the undergraduate students to publish their project. Also would like to thanks to our guide Professor S.Y.Inamdar & respected teachers for their constant support and motivation for us. Our sincere thanks to Dr.Daulatrao Aher College of Engineering for providing a strong platform to develop our skill and capabilities.

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

BIOGRAPHIES:

Mr. Sharifnawaj Y. Inamdar,
Professor, Dept. of CSE,
DACOE, Karad.

Mr. Ajit H. Jadhav,
Student, BE-CSE,
DACOE, Karad.

Mr. Rohit B. Desai,
Student, BE-CSE,
DACOE, Karad.

Mr. Pravin S. Shinde,
Student, BE-CSE,
DACOE, Karad.

Mr. Indrajeet M. Ghadage,
Student, BE-CSE,
DACOE, Karad.

Mr. Amit A. Gaikwad,
Student, BE-CSE,
DACOE, Karad.