TEST Engineering & Management

May-June 2020
ISSN: 0193-4120 Page No. 10558 - 10563

Data Integrity for Encrypted Text INHDFS

Saurabh Singhal,GLA University, Mathura, India

Deepak Mangal,GLA University, Mathura, India

Asheesh Tiwari,GLA University, Mathura, India

Article Info
Volume 83
Page Number: 10558 - 10563
Publication Issue:
May - June 2020

Article History
Article Received*: 19 November 2019*
Revised*: 27 January 2020*
Accepted*: 24 February 2020*
Publication*: 18 May 2020*

Abstract:
Big data can be defined as a large set of data which is collected from different source such as sensors, logs used for analysis. The result of this analysis is used for predicting the next step in an organization. In the world of increasing digital data, data security has become one major challenges and the rate by which data is increasing, processing the data with proper security measures shall not be ignored. The paper introduces a model for securely storing the data in HDFS by maintaining the Integrity of data.

Keywords*: HDFS, Cryptography, Storage, Integrity.*

I. INTRODUCTION

In the growing era of Digital World, the passion of general public has increased for new information technologies which has lead to exponential growth of data. Because of this the protection and security of the exchanged data has become a major issue. Indeed, from their digital nature, data can be duplicated, modified, transformation, and diffused very easily. Information Security can be defined as the protection of information and components including systems and hardware that use, store, and transmit that information [1]. However, the most critical thing is Data, whatever the type of data is, it should be held confidential.

Approx 2.5 quintillion bytes of data is generated, from which 90% of the total data is generated in last 2 to 3 years alone [2], [3]. Processing, storing and managing the data has become another big issue. Big data talks about the data sets whose volume is beyond the ability of typical database software tools to capture, store, manage and analyze [4].The most famous tool to process, store, manage, analyze is Hadoop. Hadoop was first introduced in 2005 by Doug Cutting who named it after his sons toy elephant. It has improved overtime in terms of to process and analyze the data. However, generic tools and techniques fail to provide security on Hadoop. The confidentiality of data stored on HDFS is at stake.

The Data storage for Hadoop is Hadoop Distributed File System [HDFS] [6]. Before actually putting the data on HDFS, it processes the data and divide it into chunks and store it on different nodes with transparency from the user. When Hadoop 1 was introduced, it was created with no security in mind. Hadoop 2 has some security mechanisms like for applications Kerberos RPC authentication are available; for web services, Hadoop Services, HTTP SPNEGO authentication is there, and the use of delegation tokens, block tokens, and job tokens.

Further, three encryption mechanisms for network encryption are there that must be configured Quality of Protection for SASL mechanisms, and SSL for web consoles, HDFS Data Transfer Encryption [5], [7] .However, all the data stored is in the same format as it is being input. Once attacker is inside the Data Centre either physically or electronically is free to steal, modify or delete the data they want, as there

May-June 2020
ISSN: 0193-4120 Page No. 3672 - 3680

is unencrypted data being saved and there is no authentication enforced for access [8].

A. Big Data Issues

There are various issues in Big data, which can be categorized into Processing issue [9], Storage issues, management issues and security issues [10].

1) Processing Issues: Since the size of data is enormous, usually in Petabytes, the real time processing is not always possible. The user has to opt from batch processing or stream processing.

2) Storage Issues: Storing data which is in huge volume, coming at a varied pace and is of diverse nature itself becomes a challenge. Most organization uses virtualization to hold the information.

3) Management Issues: The data in big data can be classified into structured, unstructured or semi-structured generated from various sources as public, private or government sectors. To manage this data is the biggest management issue. The purpose of the big data management is to ensure that it provides a high quality data, while maintaining data ownership, standard and accessibility.

4) Security Issues: There are various threats associated with managing big data in a secure way as the tools for the same are limited. There are chances of unexpected data leakage, fragmented data and so on. To secure data in an untrusted environment is itself difficult and if the variety is also added it become more challenging. With the varied pace and heterogeneity present in big data, it is very difficult to implement a security policy to it. to increase security of big data while at rest, some kind of framework that include cryptographic solution must be developed. In our work, we have proposed such framework.

The scheme proposed in the paper work on the model of encrypting the data before putting it on HDFS. We have analyzed different security parameters and the performance of different algorithms while encryption and decryption.

The rest of the paper is organized as follows: Section II provides the related work in the field of security in HDFS, Section III has proposed work. Section IV has result and analysis and finally conclusion is in section V.

II. RELATED WORK

In Haddop one of the important issue is data availability. Hadoop follows a master-slave structure. Therefore, if master fails the entire Hadoop cluster fails. Thus, making master node a single point of failure. In HDFS, the namenode is also a name given to the master node. The namenode or masternode, stores and manages metadata of the entire Hadoop file system.

Data availability may be lost if some errors occur happens in namenode. The provision of a secondary namenode was started from Hadoop2 to overcome the problem of single point of failure in namenode. The secondary namenode, periodically contacts primary namenode and archives the metadata present in namenode. The secondary namemode is basically designed for saving the namenodes storage space.

In [11], for Hadoop the authors had proposed a security enhancement that uses Kerberos to provide strong mutual authentication. Access control for the stored files is carried by the central server. The confidentiality of these files is always in danger as these files are stored as plain text on the storage servers. So, if the server get compromised by an attacker, the files stored in the servers are also compromised.

In [12], the author introduce a system for checking the integrity of file by considering a master slave structure of secure distributed storage system. In this distributed system, the master is introducer node and slave node is a storage node. Every user has key which is known as its signing key. When the i=user want to store a file in system, a key k1 is derived from the singing key. Then the user calculates the hash value key k2 of k1. Then the file is encrypted by using AES encryption using key k2. This double encrypted file is stored in the distributed storage system.

The Data Security model given in [10] gives the basic idea of first authenticating the user, then

May-June 2020
ISSN: 0193-4120 Page No. 3672 - 3680

encrypting the data provided by the user and securely storing the key provided to the user, the data is stored on server after encrypting it successfully.

III. Proposed Work

A. *Propsoed Scheme*

The scheme states that the data should be encrypted before storing in Hadoop Distributed File System. We have setup a Hadoop single node cluster on a Virtual Machine of Ubuntu 14.04 64-bit. For Encryption & Decryption, various cryptographic algorithms available in literature has been used. For Integrity of files, MD5 and SHA-5 are used. We have used Python PyCrypto Module for Encryption and Decryption of files. A text file is with varying sizes is taken on the local machine. The hashes of files are taken using MD5 and SHA-5 and are stored on local machine. The file is encrypted using various Cryptographic algorithm reading.

The proposed encryption model is shown in figure 1 and decryption model is shown in figure 2. The algorithm uses 128-bytes block at a time. The encrypted file is stored into Hadoop Distributed File system [HDFS]. The file is then retrieved to the local file system and decrypted using the shared / private key from the Key-pair. The hashes of the decrypted file are checked for Integrity. The hashes of the original file and of decrypted file are compared. The hashes are found to be same which means there is no data loss while encryption/decryption of files stored in HDFS. The same task is done by different file sizes and time taken by each algorithmfor varying sizes are compared. The simulation is done only for text files ranging from 1KB to 1MB of sizes.

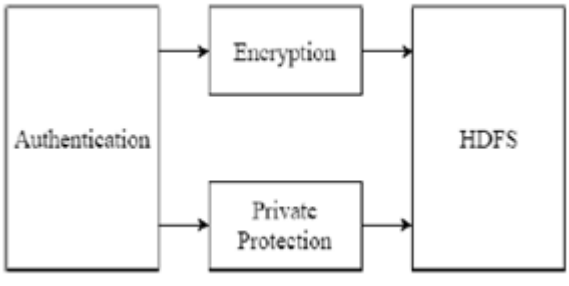

Fig. 1. Encryption model used in proposed work

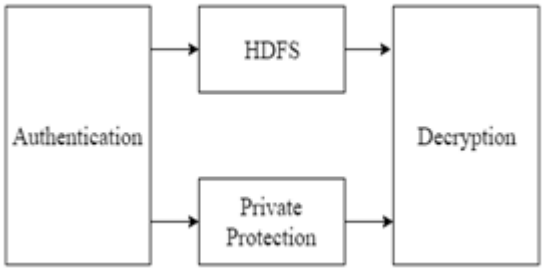

Fig. 2. Decryption model used in proposed work

IV. Performance Analysis

We have set-up a VMWare Virtual Machine of Ubuntu 14.04 64-bit on an Intel i5-4210U @1.70 - 2.40 GHz machine with 8GB of RAM and on Windows 10 Pro 64-bit. Specifications for Ubuntu Virtual Machine is 2-core CPU and 4GB of RAM. The time taken for Encryption and Decryption with varying file sizes from 1KB to 1MB are compared for four algorithms, RSA with 1024-bit key, AES with 128-bit key, DES3 with 192-bit key and Blowfish with 256-bit key.

1) Time taken in mili-seconds for Encryption with keysize in bits:

As the size of file increases encryption time per algorithm increases exponentially. The time for encryption also depends upon the size of key.

2) Time taken in mili-seconds for Decryption of Encrypted files with key-size in bits:

As the size of file increases decryption time per algorithm increases exponentially. The time for decryption also depends upon the size of key.

TABLE I. EXPERIMENTAL PARAMETERS FOR ENCRYPTION AND DECRYPTION

Entity	**Parameters**	**Values**
System	Processor	i5
	RAM	4 GB
	Storage	20 GB
	Operating Systems	Windows

TABLE II. CRYPTOGRAPHIC ALGORITHM WITH KEY SIZE

May-June 2020
ISSN: 0193-4120 Page No. 3672 - 3680

S.No	Name of Algorithm	Key Size (in bits)
1	Blowfish	256
2	AES	128
3	3-DES	192
4	RSA	1024

TABLE III. TIME TAKEN IN MILLISECONDS FOR ENCRYPTION WITH KEY-SIZE IN BITS

	Blowfish (256)	AES(128)	DES3(192)	RSA(1024)
1KB	1.195	0.151	0.463	5.598
10KB	10.258	0.865	4.519	53.449
100 KB	103.696	8.142	44.851	546.085
1MB	1085.873	84.568	447.445	5436.244

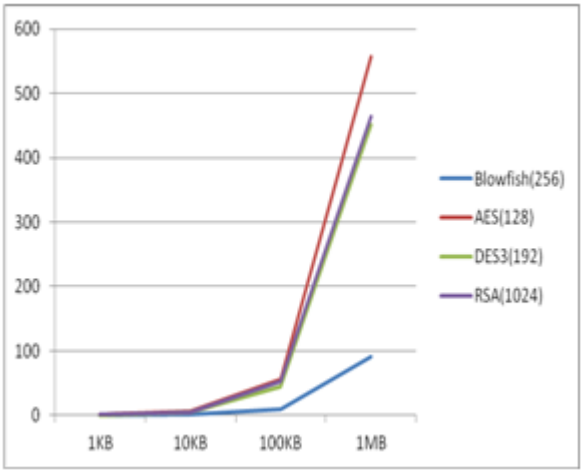

Fig. 3. Comparison of various Encryption Algorithms on test system

TABLE IV. TIME TAKEN IN MILLISECONDS FOR DECRYPTION WITH KEY-SIZE IN BITS

	Blowfish(256)	AES (128)	DES3 (192)	RSA (1024)
1KB	0.126	1.641	0.463	0.646
10KB	0.905	6.524	4.488	5.094
100 KB	9.079	55.146	44.876	52.991
1MB	90.323	556.543	450.876	463.357

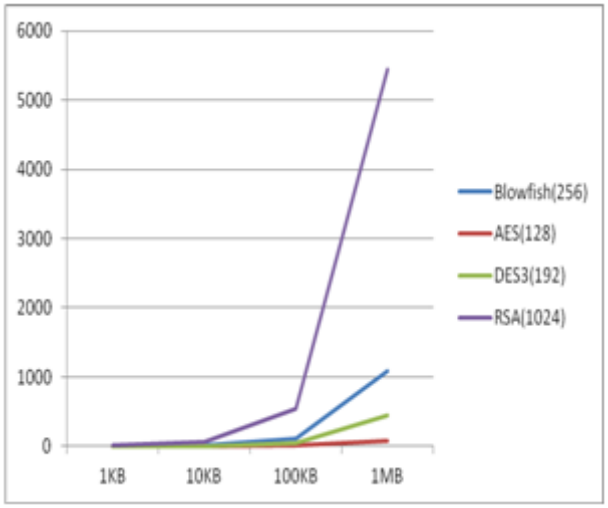

Fig. 4. Comparison of various Decryption Algorithms on test system

From the table and figures, we can deduct that time taken to encrypt and decrypt for each algorithm is constantly increasing with size of the file. However, the time taken to encrypt / decrypt large files is very high, so files over size more than 1MB are not considered here. From the measurements we have taken, we have deduced the relation which is true for 80% of the cases:

Total Time taken to encrypt = αx Size of the file (in KB) ± 5%

Total Time taken to decrypt = β x Size of the file (in KB) ± 5%,

where α, β(in secs) are for unit time in Encryption and Decryption respectively

The time taken to encrypt and decrypt files up to 1MB with high performance machines is quite low and acceptable. So, the scheme is feasible for files up to 1MB in real world. Although, Asymmetric

May-June 2020
ISSN: 0193-4120 Page No. 3672 - 3680

cryptographic algorithms like RSA is quite slow for larger files as compared to symmetric cryptographic algorithms, so it will not be feasible for larger files. However, Symmetric algorithms like AES and DES3 are quite fast and can be used for even larger files. The time taken to decrypt files with AES is low, so it will be the best choice of these four as far as time complexity is concerned.

V. Security Analysis

A. *Brute Force Attack*

More the key size, more the time taken to brute-force it. So,the order of decreasing time will be: RSA >Blowfish >DES3 >AES

B. *Avalanche Effect*

Avalanche effect of AES is highest among all the symmetric key encryption algorithms used here under both One-bit change in Plaintext and One-bit change in Key conditions [7, 8, 11]. AES >3DES >Blowfish However, Asymmetric and Symmetric Encryption algorithms are not compared in terms of Avalanche Effect.

C. *Insider Attack*

HDFS files can be viewed by anyone who has access to
Hadoop Cluster [6]. The scheme proposes that the file should be encrypted before storing it in HDFS. Even if the attacker has access to Hadoop Cluster, the file cannot be viewed by them. The attacker will only be able to see the encrypted file.

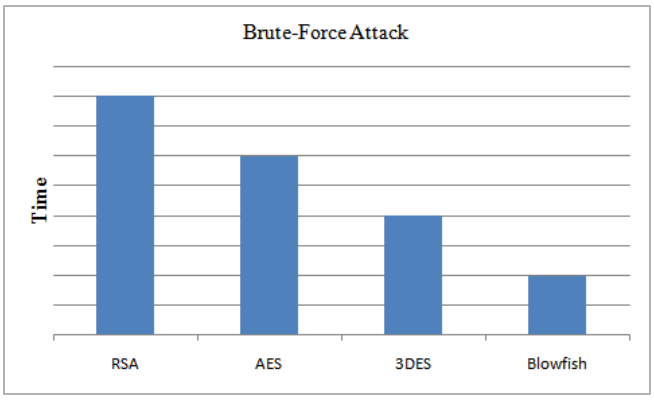

Fig. 5. Comparison of brute force attack

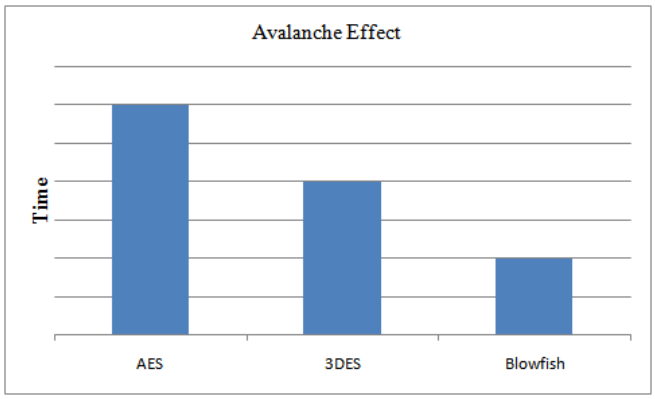

Fig. 6. Comparison of Avalanche Effect

VI. Conclusion

As the need for processing data at high speed with proper data security mechanisms. The paper discusses the model for encrypting data before storing it in HDFS, with their performance and security analysis to select one algorithm over other. The Integrity of files have been maintained while storing the files in HDFS using the scheme.

References

1. Saraladevi, B., et al. ”Big Data and Hadoop-A study in security perspective.” Procedia computer science 50 (2015): 596-601
2. Sagiroglu, Seref, and Duygu Sinanc. ”Big data: A review.” 2013 international conference on collaboration technologies and systems (CTS). IEEE, 2013.
3. Dev, Dipayan, and Ripon Patgiri. ”A survey of different technologies and recent challenges of big data.” Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics. Springer, New Delhi, 2016.
4. Vidyavathi, B. M. ”Security Challenges in Big Data.” International Journal of Advanced Research in Computer Science 6.6 (2015).
5. Smith, Kevin T. ”Big data security: The evolution of hadoops security model.” InfoQ. 2013.
6. White, Tom. Hadoop: The definitive guide. ” O’Reilly Media, Inc.”, 2012.

May-June 2020
ISSN: 0193-4120 Page No. 3672 - 3680

7. Das, Devaraj, et al. ”Adding security to apache hadoop.” Hortonworks, IBM (2011): 26-36.
8. Sharma, Priya P., and Chandrakant P. Navdeti. ”Securing big data hadoop: a review of security issues, threats and solution.” Int. J. Comput. Sci. Inf. Technol 5.2 (2014): 2126-2131.
9. Ji, Changqing, et al. ”Big data processing in cloud computing environments.” 2012 12th international symposium on pervasive systems, algorithms and networks. IEEE, 2012.
10. Song, Young-Sae. ”Storing Big Data-The Rise of the Storage Cloud.” Advanced Micro Devices, Inc. AMD (2012).
11. Russom, Philip. ”Managing big data.” TDWI Best Practices Report, TDWI Research (2013): 1-40.
12. Shehzad, Danish, et al. ”A novel hybrid encryption scheme to ensure Hadoop based cloud data security.” International Journal of Computer Science and Information Security (IJCSIS) 14.4 (2016).
13. Wilcox-O’Hearn, Zooko, and Brian Warner. ”Tahoe: the least-authority filesystem.” Proceedings of the 4th ACM international workshop on Storage security and survivability. 2008.