Performance Analysis of Implementation of AES (XTS)-MR in HDFS and Its Suitability in UIDAI

Arpana Chaturvedi[#1], Dr. Vinay Kumar[*2,] Dr. Meenu Dave[*3]
#1Asst.Prof & Department of Information Technology & Jagannath International Management School
JIMS, GGSIPU, New Delhi 110070, India
[*2] *Professor& Department of Information Technology & Vivekananda School of IT*
VIPS, GGSIPU, New Delhi 110 088, India
[*3]*Professor, Jagannath University, VIPS, GGSIPU, New Delhi 110 088, India*

Abstract

*The UIDAI data stored at CIDR is at high risk. Due to the introduction of new technologies and advanced tools, the risk factor increased. The data at rest as well as data at travel mode, both are at risk. The government is enforcing citizen of India to link Aadhaar number with different services so that right citizen can avail the benefits. The fear of getting shared private information increases the worries about various security threats. An efficient encryption technique is required to be implemented to secure data stored in different data centers. In this paper, an encryption process to be performed in parallel mode is discussed. This proposed algorithm minimizes the time consumption and vulnerabilities to numerous attacks on stored data. The proposed approach is AES in XTS Mode in Map Reduce paradigm which supports parallel programming in the distributed environment. The findings after reviews of results show that it provides better security against external attacks and overcomes the shortcomings of Kerberos. Encryption followed by compression on various datasets provides better result and protection from vulnerabilities and threats. In this paper, we reviewed different observations which prove that the algorithm is an appropriate choice.***

Keywords — *AES, XTX, Map Reduce, UIDAI, CIDR, Kerberos.*

.

I. INTRODUCTION

In last two years 80% of total data collected since epoch has increased enormously with high velocity. This extraordinary growth in data generated is due miscellaneous activities performed by common man. The advent of technologies like cloud computing, active use of social media, mobile computing, internet of things, sensor-based network etc. requires large and efficient storage with security. This advancement in technology increases the generation of data exponentially and need the strict security strategies to be implemented in the system. The sensitive data stored in various datacenters are not fully trustworthy. The government initiated the concept of Digital India and enforces end users to link their Aadhaar number with various services. UIDAI system and related infrastructure have already implemented strong security authentication and compliances. With the pace, speed and momentum the citizens are enforced to link their Aadhaar number with various services to avail benefits in that momentum only risk issues are increasing and their worries about the privacy of data increased. The citizen of India has submitted their demographic and biometric details to the UIDAI system. To avail benefits, their private and personal data need to get shared among different agencies. They need assurance that the data will not be going to get misused. Many issues in past had taken place regarding misuse of private information, leakage of private information and theft of data. IBM estimates that 90% of the world’s data was generated in last few years alone and the survey showed the major challenges identified in recent years only. Major challenges are related to reliable storage, efficient processing, data integrity and recovery. It is necessary to modify the existing security compliances, legal provisions and auditing policies. Several security issues have been identified in the current era are related to use of data at rest, confidentiality, privacy, external attacks etc. UIDAI support Hadoop Map Reduce an open source as stores the data in HDFS. There are so many technologies and algorithms available which might be appropriate to handle security issues. Various suggested technologies in different papers already written and published by me are Quantum Cryptography, quantum cryptography with steganography, Kerberos, various tools and techniques like white box testing and code Obfuscation, Homomorphic encryption, Searching symmetric encryption. The study concludes that the implementation of AES-XTX with map reduce parallel programming will be the cost-effective solution to process such a large user-generated vital and sensitive data.

II. AES - MR

The combination of Advanced Encryption Standards (AES) and Map Reduce (MR) is termed as AES-MR. It is the proposed algorithm for my research work related to the case study done for UIDAI to provide higher data security. This encryption algorithm is to provide Data level security. Map Reduce is a parallel programming language and AES is the encryption algorithm suitable for longer messages. It is suggested that the best features of both the techniques should be used together to provide much stricter security features in the introduced security layer.

Map Reduce with AES-XTS have the capability to compose applications that generates endless data during runtime. It has ability to adapt non-critical failures and can perform better planning, testing of information. In case of failures, it re-executes the failed jobs in clusters of machines.

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

A. AES-XTS -Proposed Encryption Algorithm

In this proposed technique AES is used with XTS mode. The XTS modes contains XEX-TCB-CTS (XTS) mode where XTS stands the XEX Tweakable Block Cipher with Cipher Text Stealing. It is supported by IEEE 1619-2007 standards. The XTS mode performs parallel executions and allows pipelining in respective executions. Data Encryption Standards which were used earlier is vulnerable to Brute Force attacks due to small size of key (53 to 2054 bits). US Government Agency NIST (National Institute of Standards and Technology) selected Rijndael's Algorithm as Advanced Encryption Standard. It is a better security standard which is now becoming an Industry Standard.

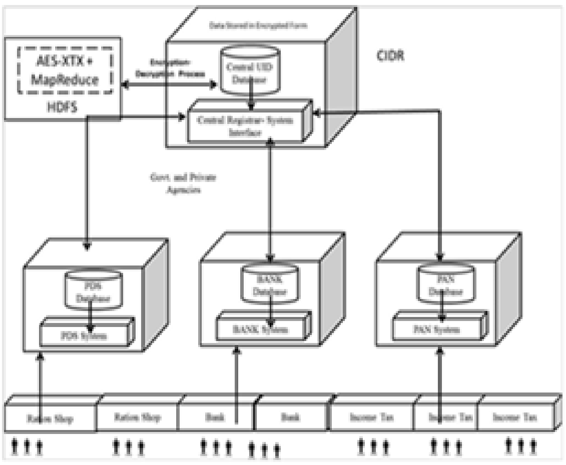

Fig..1 Model to Show Implementation of AES-XTS – MR Encryption Technique in UIDAI

AES is designed to accept various sizes of keys i.e. 128,192,256 bits. It is capable of encrypting all various types of information in bulk. The performance of AES algorithms varies on different 32 bit and 64-bit CPU's based on key sizes. This technique will provide better security (Fig.1) in case of UIDAI system where the large data sets are generated for storing sensitive information of residents and generating UID numbers of citizen of India.

B. XTS- Encryption Mode:

The efficiency of an algorithm that runs in parallel mode can be improved by the use of Electronic Cook Book (ECB) and XTX with AES. The XTX widely supports parallel encryption mode with encryption mode with Symmetric Block Cipher. It was designed to protect data lying at rest on storage devices. It performs cryptographic protection of data at rest using fixed size of data units. The operation of AES-XTX Mode with two different keys is shown in Fig. 2.

The XTS-AES mode is an enhanced concept of Rogaway's XEX (XOR Encrypt XOR) Tweakable Block Cipher, improved with a method called "Cipher Text Stealing". It expands the range of possible input data strings. XEX can only encrypt sequences of complete blocks which can be any type of data. This input data should be in integer multiple of 128 bits. In XTS-AES, the data string consists of one or more complete blocks which is followed by a single, non-empty partial block ().

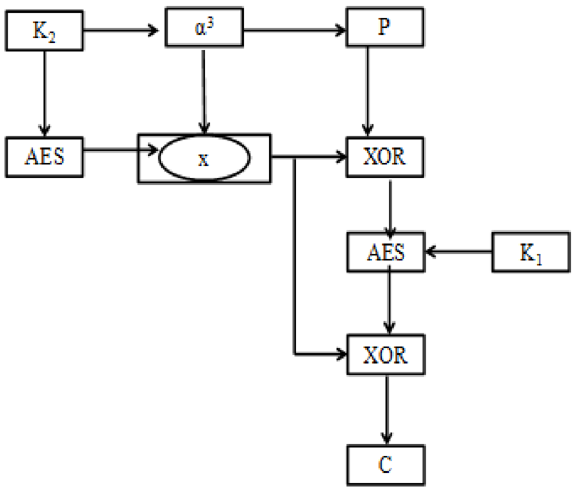

Fig.2 Operation of AES-XTX Mode with Two Different Keys

The XTX-AES consists of an encryption key and tweak key. It incorporates the logical position of the data block into the encryption. The XTS produces independent outputs which lead to parallelization. As XTS is an instantiation of the Tweakable Block Cipher class. The XTX mode allows parallelization and pipelining in cipher implementations. It enables the encryption of last incomplete block also.

III.MAP-REDUCE FRAMEWORK

A Map-Reduce computation consists of two phases, the Map phase and the Reduce phase. The map function is used in the Map phase and the reduce function is used in the reduce phase. These two user-defined functions are executed over large-scale of data. It is represented in the form of key, value pairs and also produces the key, value pairs. The working of Mapper and Reducer is shown in Fig. 3.

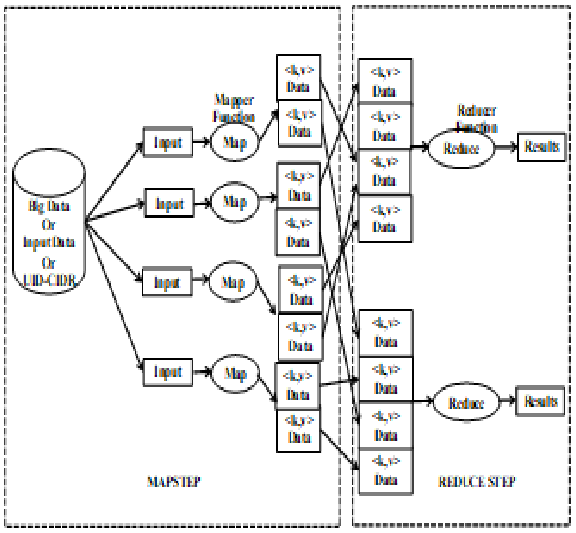

Fig. 3 Working of Mapper and Reducer

A. *The Map Phase*

The given input data is processed in the map phase, where the map function is applied to data and produces intermediate outputs in the form of (key, value) pair. In this phase it is not necessary that the number of bits required to describe the value in each key, value pair should be identical. Keys and values will always be in the form of binary strings. The Map Reduce library groups together all intermediate values associated with the same intermediate key and passes them to the reduce function. The application of the map function to a single input is called a mapper. It can be a tuple of a relational database or a node in a graph.

B. *The Reduce Phase*

The Reduce phase performs Map Reduce computations and provides the final. The Reduce Function is applied on intermediate outputs in the Reducer Phase. The application of the reduce function to a single key and its associated list of values is called a reducer

C. *Parallelizing encryption using Map Reduce Process*

The advantage of using this method is the easy and effective way to use parallelization. Each mapper μ_r operates on only one tuple at a time. The system can have many instances of μ_x operating on different tuples in μ_{r-1} in parallel. After the mapping step, the system partitions the set of tuples output throughout the instances of μ_r which are created on the basis of their key. The part of the partition i has all the key-value pairs that have key k_i. Since the reducer p_r only operates on one part of the partition, the system based on the application creates many situations of p_r running on different parts in parallel. The data will be stored as contiguous blocks and every block is represented by unique block id (I_0, I1, I2, I3…I_{n-1}). A Map-Reduce process contains mapper (M_1, M_2,M_3,…M_r) and reducer (R_1,R_2,R_3,…R_r). The input is given to the mapper in the form of <block id, object>, where an object is the content of the HDFS block or the data stored in the corresponding block id. The working of Mapper and reducer function during encryption is based on <key, value> pair and defined as:

D. *Execution of mapper*

Block< I_{n-1}, Object ds1> is given to the mapper M_r. the mapper will generate the corresponding encrypted output $I`_r$ and send it to the reducer R_r.

Let
W={<I_0,Object1>,<I_1,Object2>,<I_2,Object3>…….<I_{n-1},Object n> then general format is represented as:

$I`r=I_{r-1}$-$W_{<blockid,object>=}$Mr(<blockid,Enc_comp $_{object}$>).

E. *Execution of Reducer*

The collected outputs from various mapper are written to the disk in the sequential order ($I`_1$,$I`_2$,$I`_3$,…$I`_n$). In the proposed work the algorithm is designed for securing large datasets, It secures it by performing block encryption followed by compression under each mapper and combining the result and storing it on HDFS (Hadoop Distributed File system). The mapper reads the block of equal size. These read blocks can be optimized based on the available free nodes in the cloud environment. The working is shown below:

$$\textbf{\textit{BLOCK SIZE}} = \frac{\textbf{\textit{INPUT DATA SIZE}}}{\textbf{\textit{No. OF NODES CONFIGURED FOR MAPPER}}} \quad \text{... (1)}$$

$$\boldsymbol{B} = \frac{\boldsymbol{I}}{\boldsymbol{N}} \quad \text{…….. (2)}$$

In our case the number of Mapper configured to do the task is equal to 1.

$$\boldsymbol{No.OF\ MAPPERS\ NODES}(N) = \mathbf{1} \quad \text{…….. (3)}$$

Therefore, in this case we can say that

$$\boldsymbol{BLOCK\ SIZE\ B} = \boldsymbol{INPUTDATA\ SIZE}\ (\boldsymbol{I}) \quad \text{…..… (4)}$$

The Fig. 2 shows how HDFS contents got assigned to mapper where mapper performs encryption followed by compression and the reducer is used to write the contents onto HDFS (i.e., output). Hadoop

Distributed File System (HDFS) supports record level and block level compression of input data. In the proposed method in order to reduce the storage space, after encryption under each mapper, a compression through gzip is done.

IV. EXECUTION OF AES – MR

A. *Encryption*

AES-MR is used to encrypt the data stored in data store. The steps in performing Encryption as shown in Fig. 4. are:

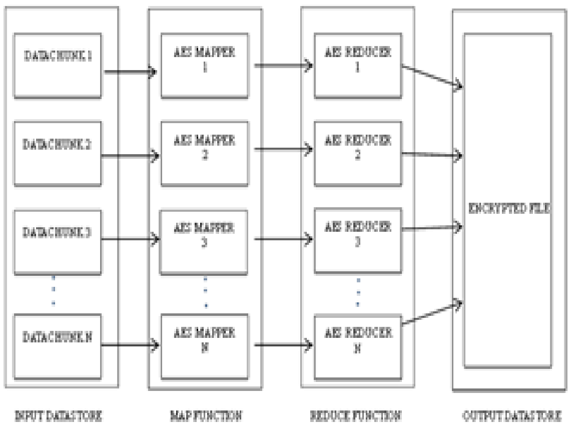

Fig 4. Encryption Process in Map Reduce

- The data are taken from the Data store in the form of chunks of fixed sizes.
- These chunks are then forwarded to the Map Reduce for the process of encryption.
- The Map function contains the code for the encryption, AES in XTX mode. It encrypts the data chunk by chunk in parallel and converts the chunks into encrypted chunks.
- These encrypted chunks are forwarded to the Reducer function of the Reduce phase. It combines all the encrypted chunks in a single encrypted file.
- This single encrypted file of the original file is stored in the Data store and the process of encryption is completed.
-

B. *Decryption*

Decryption process is opposite of encryption. It is done using AES-MR, where it performs decryption using encrypted file stored in the data store. The steps used in performing decryption process as shown in Fig. 5 are:

- The input the decryption phases are taken from the Data store in the Encrypted format.
- This encrypted file is broken into chunks and then forwarded to the Map Reduce Functions Mapper class
- The Map function of the Mapper class contains the decryption code. It decrypts the encrypted chunks one by one in parallel and converts it to plain data format.
- These unencrypted data chunks are then forwarded to the Reducer function of the Reduce class. It sums up and combines the unencrypted simple data chunks into a single unencrypted file.
- This single unencrypted file is again stored in the data store and can be viewed easily.

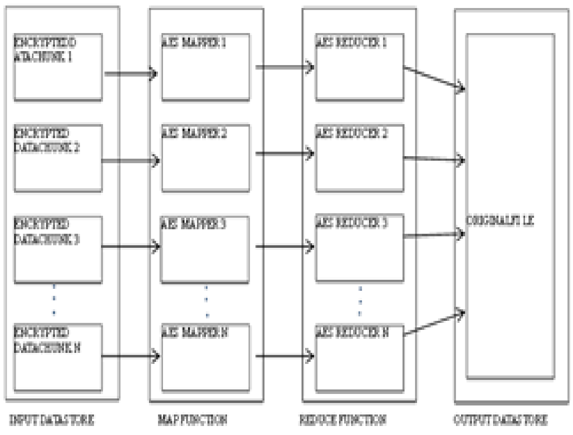

Fig 5. Decryption Process in Map Reduce

The process of decryption phases is the opposite process of encryption phase. The encrypted data is divided into the encrypted data chunks and forwarded to the Map Reduce phase. It decrypts the data into simple file and then stores it into the datacentre.
The conclusions drawn after experiments performed demonstrate the performance of encryption. It has been observed that the encryption performance can be significantly improved by tuning configuration parameters and by using Map Reduce. The parallel processing improves encryption performance in terms of total execution time. We evaluated the execution time of AES encryption for different sizes of data files using Map Reduce with tuning configuration parameters. Execution time is compared to those obtained using CTR AES-256 in the conventional way. According to the results there are 20% to 30% performance benefits because of the use of the Map Reduce framework. Two reducers are used for the test. The mapper needs to spend more time to partition jobs as the multiple reducers are used. Each of the reducers has only some part of the encrypted file and the time of the reducers can decrease due to the parameter settings.

V. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results and benefits of this proposed algorithm done using simulation environment to evaluate the performance is reassembled together in this paper.

A. First Simulation Environment:

The environment used is MATLAB r2015, Intel® Core™ i3 CPU M 370@2.40 GHz, 4 GB RAM and 64 Bit Windows 7 Home. The experimented results are drawn are done using various randomly created sizes of datasets. The experiments are carried out to calculate the average values of the parameters evaluated for performance and comparisons.

1) *Parameters used for evaluation of performance*

The overall security architecture of a Hadoop application required to maintain security at various stages. The strongly preferred essential levels of security are Authentication, Authorization Auditing and Data Encryption. The 3ADE level (Fig. 6) of security when implemented against external attacks is depicted as:

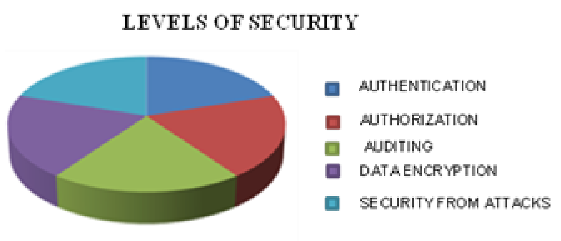

Fig 6. Various Essential Security Levels

2) *Limitations* OF KERBEROS PROTOCOL

With the help of pie chart (Fig. 7), it is clear that except Authentication and Authorization provided by Kerberos and other security protocol, rest of the equally important security levels are not considered.

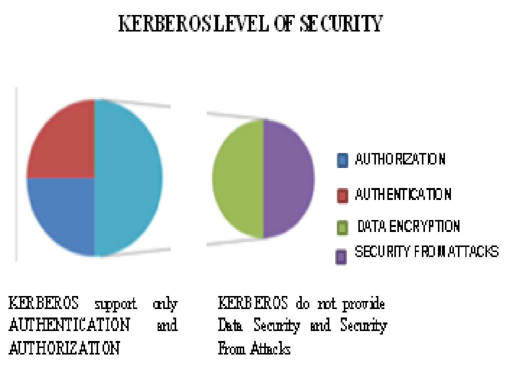

Fig. 7. Limitation of Kerberos Authentication Protocol

3) *Experimental Results*

The performance of AES-MR is measured on the basis of Encryption Time, Decryption Time and Total Time taken by this algorithm to encrypt and decrypt the data. The performance is shown with the help of pie chart including the security enhancement done by implementing AES-MR. The result has been shown with the help of pie chart, the shortcoming of Kerberos Protocol and explains the impact Before AES-MR and After AES-MR.

a) AES-MR Data Encryption Time:

To encrypt the data completely, the data from the Data stores is supplied to AES-MR. The Mapper function is used to encrypt the chunks in parallel and then these encrypted chunks are forwarded to Reducer function. The reducer function combines together all the encrypted chunks. It gives the final encrypted output for the data sent. The Total AES Encryption Time taken in this process is the total elapsed time used to pick the data from the data store and the time taken to store the encrypted data back into the data store. For experiment purpose the various size of data files which are taken are of 1 MB, 25 MB, 50 MB, 75 MB and 100 MB size. The graph (Fig. 8) shows the encryption time taken by AES-MR to perform encryption on various sizes of data.

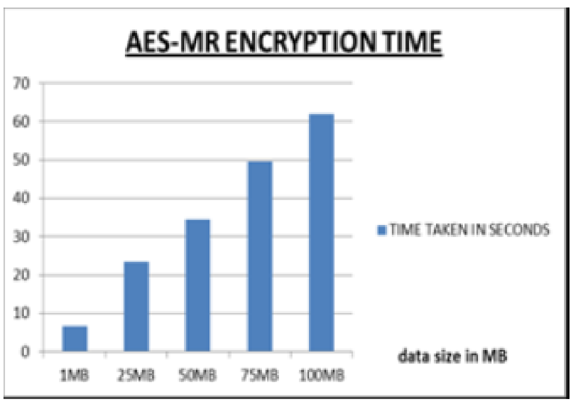

Fig. 8. Total Encryption Time Taken by AES-MR

b) AES-MR Data Decryption Time:

To decrypt the data completely, the encrypted data from the Data stores is picked and is supplied to AES-MR. The Mapper function is used to decrypt the chunks in parallel and then these decrypted chunks are forwarded to Reducer function. The reducer function combines together all the decrypted chunks and produces original Decrypted File. It gives the final encrypted output for the data sent.

The Total AES Decryption Time taken in this process is the total elapsed time used to pick the encrypted data from the data store and the time taken to store the decrypted original data back into the data store. For Experiment purpose the various size of data files are taken the sizes are 1 MB, 25 MB, 50 MB, 75 MB and 100 MB. The graph (Fig. 9) shows the Decryption time taken by AES-MR to perform decryption on various sizes of encrypted data.

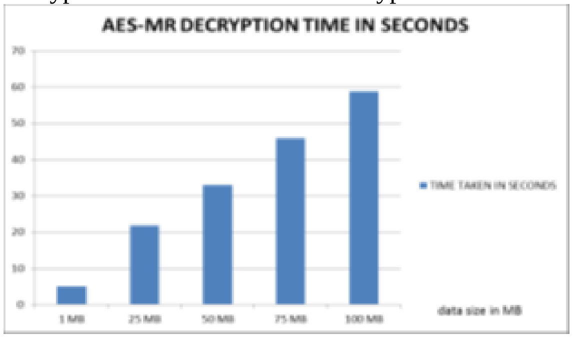

Fig.9: Total decryption Taken by AES-MR to Decrypt the Data

c) Overall Performance of AES-MR:

The overall performance of AES-MR is measured by the summing up the overall time i.e. the total time taken during the entire process to perform all operations. The table (Table 1) shows the particulars on the basis of which the overall performance is evaluated and depicted in the form of graph.

DATA SIZE IN MB	TIME TAKEN FOR ENCRYPTION	TIME TAKEN FOR DECRYPTION	TOTAL TIME TAKEN
1 MB	6.79	5.06	11.85
25 MB	23.48	21.85	45.33
50 MB	34.56	32.89	67.45
75 MB	49.59	45.82	95.41
100 MB	62.04	58.75	120.79

Table 1: Overall Performance Evaluation of AES-MR on the basis of Time Taken

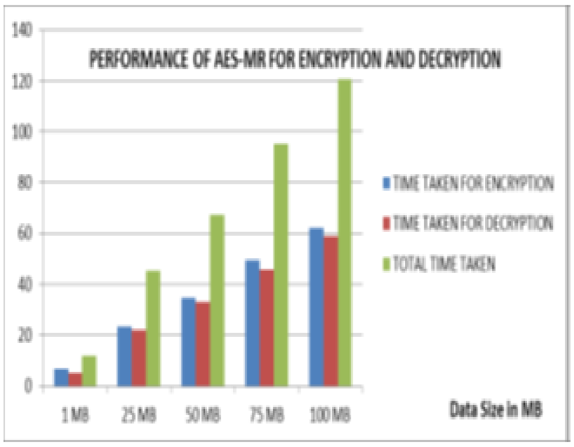

Fig. 10: Overall Performance Evaluation of AES-MR

d) Evaluation of Security Enhancement Using AES-MR

The usage of AES in XTX mode with Map Reduce function for encryption of data chunks increases the level of security. It is the more secure and fast method of encrypting vital data stored in the Hadoop Data store. The encryption and security against the attacker was not provided by Kerberos protocol. AES-MR overcomes the limitation of Kerberos by providing the security against attackers and enables implementation of the four important security levels i.e. Authentication, Authorization, Auditing and Data Encryption as shown in graph (Fig. 11).

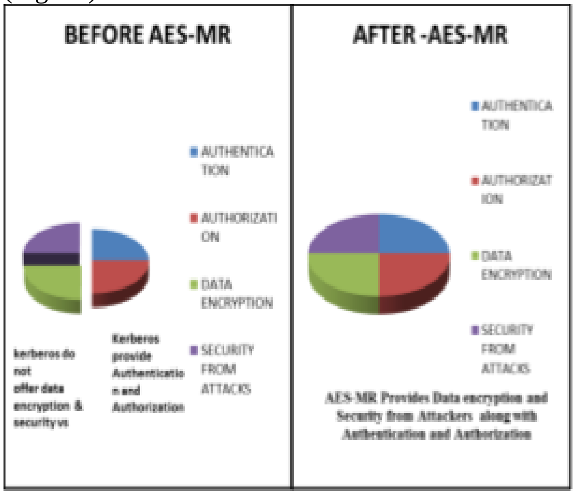

Fig. 11: Security Enhancement Before and After AES-MR

B. Second Simulation Environment:

The NS2 simulator is used which consists of a 28 node having 4 cluster, each of which has an Intel® Core™ i3 CPU M 370@2.40 GHz, 4 GB RAM and 64 Bit Windows 7 Home. It is being utilized to perform encryption followed by compression while storing different datasets.

1) Mode Comparison that supports parallelism with AES:

The experimentation is done with AES Encryption with ECB Mode and XTX Mode and time taken by both in performing encryption of vital data stored in Hadoop data store is evaluated. As a result, it is found that the time difference in both for protecting data is very less whereas the output produced by both for the same data is different. Different mapper is used and it has been concluded that when AES is used in XTS mode for encryption it produces a different output for the same data and when used in EBC mode, produces same cipher output for the same data. The difference in time taken for encryption of 15 GB of data is 3 minutes (Fig. 12) concludes that usage of AES in XTS mode is comparatively much secure of larger datasets stores in the form of clusters in Hadoop.

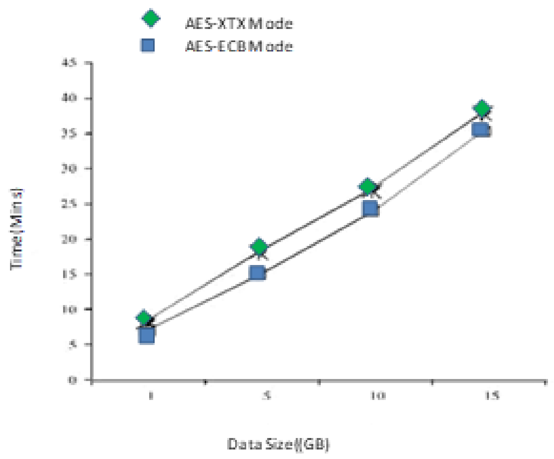

Fig. 12: Time Taken For Encryption in AES-XTS and AES-ECB Mode

2) ***Performance of AES (XTX)-MR with reducer and Without Reducer***:

In this experimentation it has been evaluated that when Map Reduce algorithm is used with AES in XTX mode with reducer, all the nodes of the cluster were assigned as mapper. It increases the speed of encryption of the vital data stored in Hadoop clusters. The performance of AES-MR with reducer and without reducer is shown in Fig. 13. This encryption process is further followed by compression process also.

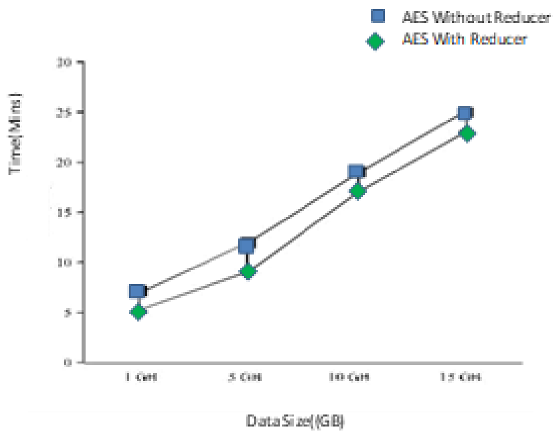

Fig. 13: Performance Of AES-MR with Reducer and Without Reducer

3) ***Performance evaluation of AES (XTX) encryption with Map Reduce Process for different Datasets:***

It has been observed after experimentation that the time taken for encryption applied for different larger size datasets different data types i.e. text, audio, video and image is different. The time taken for encrypting text file is more than the time taken for encryption of video, audio and image files. It has also been observed that in the encryption of audio and video files are almost same. The evaluation of performance of AES9XTX) with different data sets is shown in Fig. 14.

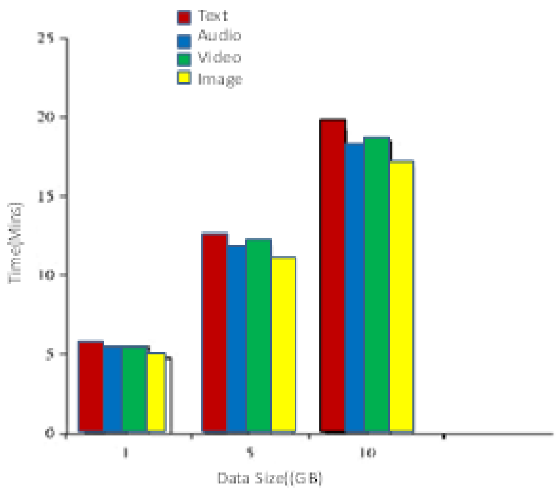

Fig. 14.: Evaluation of Performance of AES (XTX) with Different Datasets

4) ***Performance Evaluation of Compression on different datasets without Reducer***:

The compression is done on the large datasets where all the nodes of the cluster are used as a Mapper. The performance is evaluated in this without using reducer and it has been observed that the encryption with compression takes less time. (Fig. 15) It has also been observed that the compression ratio for text and image data sets is 1: 10 and 1: 2 whereas there is not much difference in the ratio of audio and video files.

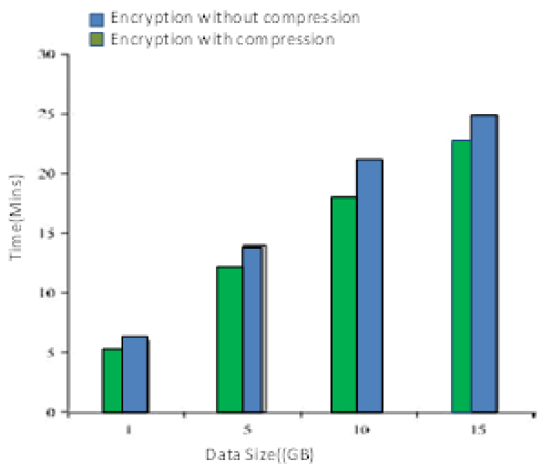

Fig. 15: Performance Evaluation of Compression on Different Datasets Without Reducer.

VI. CONCLUSION

In this work we tried to solve this issue of data security at the storage level at HDFS by encrypting the data using AES-MR, where we have combined AES the fastest encryption algorithm with Map Reduce in order to encrypt the data in a faster manner. The timing graphs show that the AES-MR encryption technique is fast enough to encrypt the data chunks from the data store using parallel processing of map and reduce functions. Thus with the use of AES-MR we have not only enhanced the security of important data at HDFS level, but with the help of parallel processing we can do it with a faster manner as well. The motive of this work was to ensure the security of important data at an HDFS storage level which has not been achieved by Kerberos only, but with our work along with Kerberos ensures security at each and every level and makes it almost impossible for the attacker to get the data which has been achieved in our work. In our simulation the work is carried out by a single worker this work can be done with even more speed if we increase the number of workers which is possible in the real world. The chunk sizes can also be distributed with the use of large numbers of workers, which in our case was the data size itself. Overall, if this work gives the data security a new height along with the use of Kerberos to attain all the levels of security that too at a faster speed, AES-MR is a good approach to carry out his work and attain data security.

ACKNOWLEDGEMENT

This research was supported by various research work done by researchers and published papers. I thank all of them as their research work has provided8 the insight and expertise to me and their work has greatly assisted my research. I would like to thank and show my gratitude to all reviewers for sharing their pearls of wisdom with me during the course writing this research paper. I am immensely grateful to the reviewers for their comments on an earlier version of the manuscript, although any errors are my own and should not tarnish the reputations of these esteemed persons.

[2] REFERENCES

[1] Kadre V., Chaturvedi S., “AES – MR: A Novel Encryption Scheme for securing Data in HDFS Environment using Map Reduce”, www.ijcaonline.org/ research/ volume129/ number12/kadre-2015-ijca-906994.pdf

[2] Mehak, Gagan, “Improving Data Storage Security in Cloud using Hadoop “, ISSN: 2248-9622, Vol. 4, Issue 9(Version 3), September 2014, pp.133-138, http://www.ijera.com/papers/Vol4_issue9/Version%203/U4903133138.pdf

[3] Weeks B., Bean M., Rozylowicz T., Ficke C.,” Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms, National Security”, https:// csrc.nist.gov/CSRC/ media/Projects/Cryptographic-Standards-and-Guidelines/documents/aes-development/NSA-AESfinalreport.pdf doi=10.1.1.35.6941

[4] Clunie D., Public Comments on the XTS-AES Mode,” https://csrc.nist.gov/csrc/media/projects/.../comments/xts/collected_xts_comments.pdf

[5] Public Comments-Modes Development - Block CipherTechniques,"https://csrc.nist.gov/Projects/Block-Cipher-Techniques/BCM/Public-Comments-Modes-Development “, Comments submitted to EncryptionModes@nist.gov.

[6] Dr. Hawthorne, NY, Computing Arbitrary Functions of Encrypted Data Craig Gentry IBM T.J. Watson Research Center 19 Skyline,cbgentry@us.ibm.com https://crypto.stanford.edu/craig/easy-fhe.pdf.

[7] Desai Spark Y., Gao J., Sang-Yoon Chang, Chungsik Song, “Improving Encryption Performance Using Map reduce”, Published in: High Performance Computing and Communications (HPCC), IEEE 17th International Conference, ISBN: 978-1-4799-8937-9, http:// ieeexplore. ieee.org / document/7336355/

[8] G. Sujitha, M. Varadharajan, B. Raj Kumar and S. Mercy Shalinie ,”Provisioning Mapreduce for Improving Security of Cloud Data “,http:// scialert.net/ qredirect.php? doi = jai.2013.220.228& linkid=pdf, International Journal of Computer Science and Applications, © Technomathematics Research Foundation Vol. 13, No. 2, pp. 89 – 105, 2016.

[9] Alexander Uskov, Adam Byerly, Colleen Heinemann,” Advanced Encryption Standard Analysis with Multimedia Data on Intel® AES-NI Architecture”, Department of Computer Science and Information Systems, and interlabs Research Institute Bradley University, 1501 West Bradley Avenue Peoria, Illinois 61625, U.S.A. auskov@bradley.edu http://www.tmrfindia.org/ijcsa/v13i26.pdf

[10] Demir L., Thiery M., Roca V., Jean-Louis Roch, Jean-Michel Tenkes, “Improving dm-crypt performance for XTS-AES mode through extended requests “, Nov 21, 2016 The 4th International Symposium on Research in Grey-Hat Hacking - aka GreHack, Nov 2016, Grenoble, France https://hal.inria.fr/hal-01399967

[11] Philip Derbeko, Shlomi Dolev, Ehud Gudes, Shantanu Sharma” Security and Privacy Aspects in map reduce on Clouds: A Survey”, www.https://arxiv.org/abs/1605.00677, www.iiste.org, ISSN 2224-610X (Paper) ISSN 2225-0603 (Online) Vol 2, No.2, 2012

[12] Liskov M., Mine Matsu K., “Comments on XTS-AES” September 2, 2008 This is a comment in response to the request for comment on XTS-AES, as specified in IEEE Std. 1619-2007 September 2, 2008, https://csrc.nist.gov/csrc/media/ projects /block-cipher-techniques/ documents/ bcm/ comm. ents/xts/xts_comments-liskov_minematsu .pdf.

[13] Kirat Pal Singh, Shiwani,” An Efficient Hardware design and Implementation of Advanced Encryption Standard (AES) Algorithm “, https :// eprint.iacr.org/ 2016/ 789.pdf.

[14] Vaidyaa M., Dr Shrinivas Deshpandeb ,” Study of Performance Parameters on Distributed File Systems using map reduce “, www.sciencedirect.com International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, https://ac.els-cdn.com/S1877050916000399/1-s2.0-S1877050916000399-main.pdf ?_tid=78c69a4a-e233-11e7-8cfd-00000aab0f01& acdnat=1513409815_ d18af66cf2c2e5fa578411397b06ce28