International Journal of Computer Trends and Technology (IJCTT) - volume4 Issue5–May 2013

Securing User's Data in HDFS

Hetalben Gajjar
Department of Computer Science,
IT Systems and Network Security,
Gujarat Technological University,
India

Abstract— With the advent of Technology and increasing growth in volume of data the business are finding the cloud as suitable option to host their data. However, putting the sensitive data on third-party infrastructure poses several security risks to their data utilizing the advantages of the Clouds. Though there are many risks and concerns are involved in cloud computing according many surveys conducted by different organizations the prime concern of clients when opting for cloud solution is the security of their data. Key issue is to protect important data from unauthorized access by adversaries in case the confidentiality of data is broken by internal or external attacks on the cloud hosting those data. HDFS is the file system suitable for storing and processing large volume of data using MapReduce model. When public cloud is based on the Hadoop which uses HDFS to store data, the data are stored in plain text and by default the transport of data is also insecure when client submit the data to storage servers on cloud. Requirement here is design and implement a prototype to secure the HDFS to harness is with security features so that it can be deployed in public cloud to provide storage and computing services. We have proposed and implemented secure HDFS by incorporating Elliptic Curve Integrated Encryption which provides data confidentiality as well as integrity in Hadoop. Experiments were carried out to analyze the performance with respect to other hybrid encryption schemes.

Keywords- HDFS, Cloud, Security, ECIES

I. INTRODUCTION

Cloud computing is expected to be the platform for next generation computing, in which users carry thin clients such as smart phones while storing most of their data in the cloud and submitting computing tasks to the cloud. A web browser serves as the interface between clients and the cloud. One of the main drivers for the interest in cloud computing is cost and reliability. As personal computers and their OS and software are becoming more and more complex, the installation, configuration, update, and removal of such computer systems require a significant amount of intervention time from the users or system managers. Instead, outsourcing the computation tasks eliminates most of such concerns. The cloud provides such facilities on demand based, which can be more cost-efficient for the users than purchasing, maintaining, and upgrading powerful servers. However current cloud computing environment poses serious limitation to protecting User's data confidentiality. There is always threat of unauthorized disclosure of data that is sensitive for user by service provider.

For a cloud consumer to deploy their sensitive data on service provider's storage infrastructures requires certain security assurances. The storage of user's data on cloud needs to consider threats not only from outside attacker but also from service provider itself that is generally following multi-tenant system where cloud infrastructure is shared by multiple consumers. There have been various security architectures for cloud computing proposed by researchers which are discussed in this survey. Also for storage intensive applications when large data sets used when stored on cloud based on HDFS which stores data in plain text, the concern is to provide security in the distributed file system. If Hadoop cluster is deployed as private cloud it can be run behind Firewall to defense the security risks, however if it is on public cloud it is not secure. So various techniques used by different DFS for security purpose are also surveyed. There are two major concerns when it comes to security in a Distributed File System. 1) Secure Communication between client and DFS 2) Secure File Storage. When the option for data encryption is considered for providing confidentiality over insecure communication channel, the selection of appropriate encryption algorithm is also important. Since key management is cumbersome for symmetric encryption more appropriate approach is to combine it with Public Key algorithm. According to the literature survey most of the public key implementation used RSA as a public key encryption or Deffie–Hellman which is susceptible to MITM. One more public key cryptography mechanism is available, that is Elliptic Curve Cryptography (ECC). Compared to RSA, the prevalent public-key scheme of the Internet today, Elliptic Curve Cryptography (ECC) offers smaller key sizes, faster computation, as well as memory, energy and bandwidth savings and is thus better suited for client that has to perform the key generation and encryption.[16,17] .Where as the encryption schemes addresses the first concern in security of a DFS the integrity of stored files can be provided by message authentication code. The Elliptic Curve Integrated Encryption Scheme [20] can be used as an option to provide both confidentiality as well as integrity check.

Another issues and challenges are key management and performance overhead when security is implemented using cryptography which is discussed later.

II. RELATED WORK

Kerberos is the considered suitable option for providing security in DFS. Kerberos is suggested as security alternative for Hadoop.[17] Kerberos is based KDC with two parts Authentication Server and Ticket Granting Server. KDC generates a session key for communication between two trusted entities. This approach can be adopted for HDFS. For integrity purpose cryptographic hash or digest for every file. However it is proposed for mutual authentication and controlling access to the data stored on nodes. Kerberos works as follows in HDFS: Instead of client sending password to application server: Request Ticket from authentication server Ticket and encrypted request sent to application server. Using TicketGrantingTicket(TGT) tickets can be requested without repeatedly sending credentials. It solves the issue of authentication but confidentiality and integrity still remains questionable.

Another solution exists in Hadoop is Tahoe-LAFS using which the secure distributed file system Tahoe is supported by Hadoop. Tahoe-LAFS can be used to store files in encrypted from in the Hadoop cluster and Tahoe can be used as alternative DFS. However it is not default File System on Hadoop and for multiple files by a user key management becomes tedious task. Tahoe is a Secure distributed File System with least-Authority. It has master-slave Architecture. Tahoe uses the capability access control model [18] to manage access to files and directories. In Tahoe, a capability is a short string of bits which uniquely identifies one file or directory. Each immutable file has two capabilities associated with it, a read capability or read-cap for short, which identifies the immutable file and grants the ability to read its content, and a verify capability or verify-cap, which identifies the immutable file and grants the ability to check its integrity but not to read its contents. For mutable files, there are three capabilities, the read-write-cap, the read-only-cap, and the verify-cap. Users who have access to a file or directory can delegate that access to other users simply by sharing the capability. Users can also derive a verify-cap from a read-cap, or derive a read-only-cap from a read-write-cap. This is called diminishing a capability. The limitation in using Tahoe with Hadoop is it becomes tedious job to manage multiple keys as number of user files increases.

Some traditional and popular Distributed File Systems [23,24,25,26] which are cluster based studied for literature survey in order to understand the security provisions they offer. The focus was on the mechanism used by those Distributed File Systems in order to provide Authentication, Authorization, Integrity and Confidentiality. Most of the Distributed File Systems uses Kerberos or password based authentication and UNIX based ACL for authorization. Data integrity check is not considered major issue and if all considered then using checksum is used. Providing confidentiality encryption is used. Various architectures and schemes[6,7,8,9,13] are proposed by researchers for data security in Cloud and also for distributed file systems which are discussed here. Most the schemes incorporate the public key and symmetric key cryptography to solve the purpose. The techniques differ in the way the number of keys are used, stored and generated. So main concern is keeping the scheme simple still robust and effective. However few of them focus on HDFS security and performance overhead analysis. Hsiao-Ying Lin et al.[19] have proposed and implemented Hybrid Encryption Scheme for HDFS using FUSE which mounts HDFS in user's space. However their work focuses on Data confidentiality and data integrity concern still remains to be addressed.

III. SYSTEM DESIGN AND IMPLEMENTATION DETAILS

For securing the user's data on Hadoop we have implemented client programs which are equivalent of fuse-dfs module provided by Hadoop in order to perform file read and write and other operations related to HDFS File system. However we have integrated the security within read and writes operations itself. That is the encryption of file contents is incorporated in write function and the process of decryption of file is embedded into read operation. The detailed flow of the steps performed is as follows. When user needs to write file to HDFS a new random secret key *sk* and initialization vector for symmetric encryption algorithm AES is generated for the file in schemes first and second. Then the *sk* is encrypted using user's public key *pk* of the asymmetric encryption algorithm and written to a separate file with same name as the file to be written to HDFS. This will relieve the user from remembering and managing secret keys. Then the contents of file are read from local file system and encrypted using *sk* and written to HDFS. AES is used in CBC mode. Similarly when a file is to be read from HDFS first the encrypted secret key associated with the file are read from HDFS and decrypted using user's asymmetric private key. Then the encrypted file contents are read from HDFS and decrypted using the retrieved secret key.

In third scheme that is ECIES the steps followed while writing a file to HDFS and reading a file from HDFS are shown in figures 1 and 2. In order to have better comparison of performance among different schemes AES with CBC mode with key size 128 is used. For integrity purpose the HmacSHA1 is used which produces message digest of size 160 bits. A 160-bit elliptic curve is used as a 160-bit key as secure as with RSA using a 1024-bit key[20]. In this scheme user just need to specify

the file to be written and the public key for ECC. And steps are carried out as shown in figure and encrypted file contents along with ephemeral public key *U* that is generated by Key Derivation Function and tag which is computed message digest of the file are written to HDFS. When the file is to be read from HDFS with ECIE first the ephemeral public key is read from beginning of file then from retrieved key *U* and user's Private Key, the symmetric key K_{ENC} and MAC key K_{MAC} for the encrypted file are generated. Also the MAC for encrypted file contents are calculated and compared with tag value retrieved from the file. If they match then the file is guaranteed to be unaltered and using K_{ENC} file is decrypted and stored on local system. For user to create asymmetric key pair of Public key and Private in case of all three schemes key generation programs are implemented. The user is supposed to generate public private key pair by running the key generation programs. The key pair is generated and stored locally so later it can be easily retrieved when required. The replication is used by Hadoop in HDFS to provide robustness. That is multiple instances of files are stored in HDFS depending upon the configured value of replication. For implementing the client program to interact with HDFS Java API provided by Hadoop is used. For RSA and symmetric encryption algorithm AES Sun Java 6 is used. However for ECC and ECIE third party provider that FlexiProvider is used.

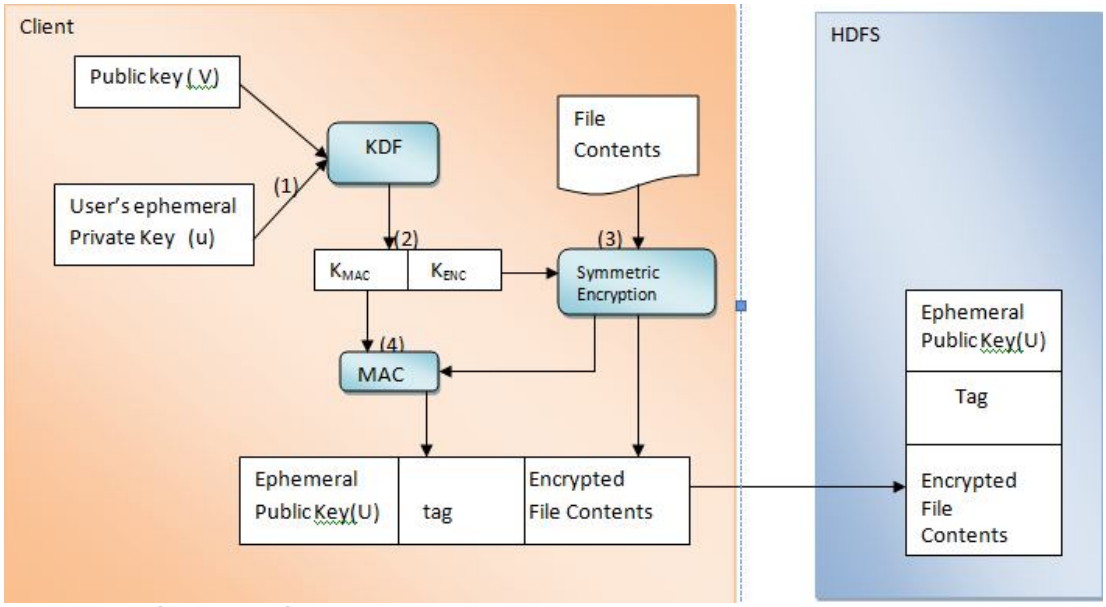

Fig 1. Writing File to HDFS with ECIE

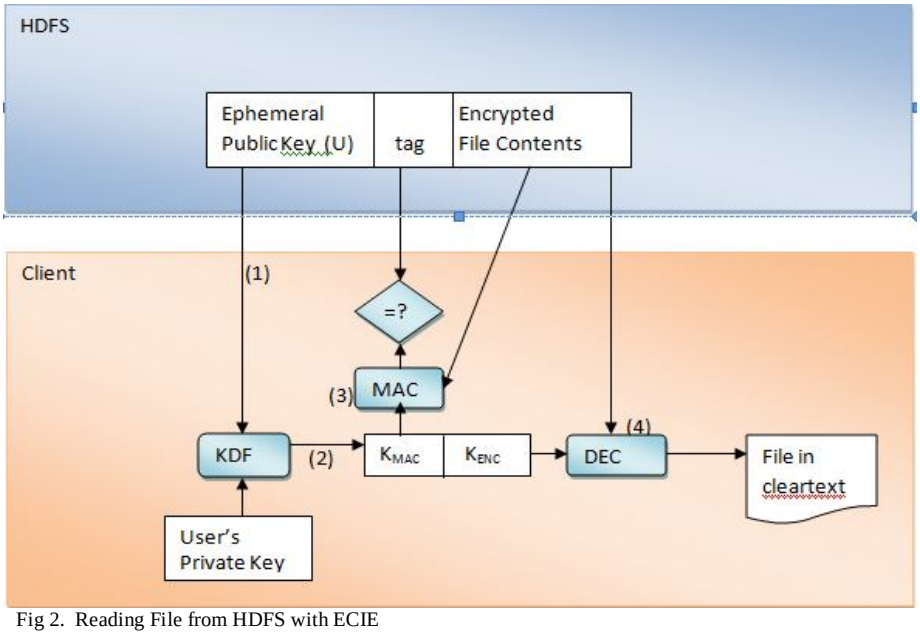

Fig 2. Reading File from HDFS with ECIE

TABLE I

EXPERIMENTAL RESULTS OF WRITING FILE WITH DIFFERENT SCHEMES

Writing time in ms								
File size in KB	RSA		ECC		ECIE		HDFS Default	
	rep=2	rep=3	rep=2	rep=3	rep=2	rep=3	rep=2	rep=3
2	818.50	831.00	582.80	667.33	699.30	562.11	897.00	730.80
94	790.60	964.56	693.10	652.67	579.40	571.00	836.11	837.20
463	931.90	1004.00	659.00	716.89	781.60	504.89	783.44	801.10
1537	1323.00	1173.00	745.00	860.22	853.80	681.89	818.89	797.90
2547	1534.50	2608.00	844.00	1161.44	1235.20	777.33	1360.56	911.70
6630	2687.00	3913.67	1244.80	2100.11	2295.10	1253.22	1204.89	961.20
9911	3486.80	4539.56	1308.30	2446.11	2729.70	1512.11	1681.67	1765.50
12326	4444.90	6489.11	1366.20	2906.11	2994.50	1735.00	1859.67	1436.10
21428	6650.10	9490.78	1984.30	4586.11	4453.20	2678.00	2503.67	1931.30
27191	8445.40	11879.22	2481.30	5763.11	5405.00	3139.67	2879.44	2404.30

IV. PERFORMANCE EVALUATION, RESULTS AND ANALYSIS

To evaluate and analyze the performance and storage overhead that is introduced by incorporating the encryption schemes into HDFS various experiments were executed. The reading and writing speeds of the default HDFS, HDFS-with combination of RSA and AES, HDFS-with combination of ECC and AES and HDFS-with ECIE is then measured. For each scheme the storage overhead is also analyzed.

A. Scenarios for Experiments

Basically we created a cluster of three nodes with hardware and software configurations mentioned in next section. Out of three nodes one acts as NameNode, Secondary NameNode and DataNode itself and the other two of the three are given roles of DataNodes only. So all together we have three DataNodes. In Scenario 1 the configuration of replication factor is 2 that is two replicas are stored on DataNodes in HDFS for each file. While in scenario 2 the set up is same but the replication factor is set to 3 so that three instances of each file are stored on the DataNodes which makes it more robust. We have not

considered the replication factor 1 as only single instance of each file is stored so the system will not be considered reliable.

The time taken to write a file to HDFS in all the four cases is measured by copying file from local system that is running as a Virtual Machine on same host to HDFS. Similarly the read time is measured in all four cases by copying the files stored on HDFS to local system. In order to get average write and read times we used 10 different files of different types with file size ranging from 2 KB to 27191 KB. In order to get more precise results each

experiment is conducted 10 times and the average of the results is taken.

B. Set up of Experiments

The experiments for analysis purpose are conducted on virtual-network in VMWare. Each virtual machine in VMware acts as one physical node in our experiment profile. All virtual machines representing physical nodes are created and run in single host machine that has hardware configuration of Intel core i53210M 2.5GHz with Turbo Boost up to 3.1GHz, 4GB DDR3 memory and 500 GB HDD.

The hardware configurations of all the virtual machines are all same i.e 512 MB memory and 20 GB disk. The operating system on each node is Ubuntu LTS 10.04.4 and Hadoop 1.0.4.

C. Write performance analysis

The table I depicts the experiment results that are writing times in each of four cases with replication factors 2 and 3. In general the time consumption in each case in each scenario gradually increases with the increase in file size.

As shown in figure 3 the performance of ECC is quite close to default HDFS. Even though there is overhead of three operations namely a) encryption of the symmetric encryption key for the file b) Writing of the encrypted key in separate file c) Encryption of the File contents is involved. Highest time is taken by RSA with AES. And ECIE that provides integrity as well by including MAC takes more time than ECC but still performs better than RSA. The graph rises rapidly in case of RSA with increase in file size while it tends to get stable in case ECIE and ECC.

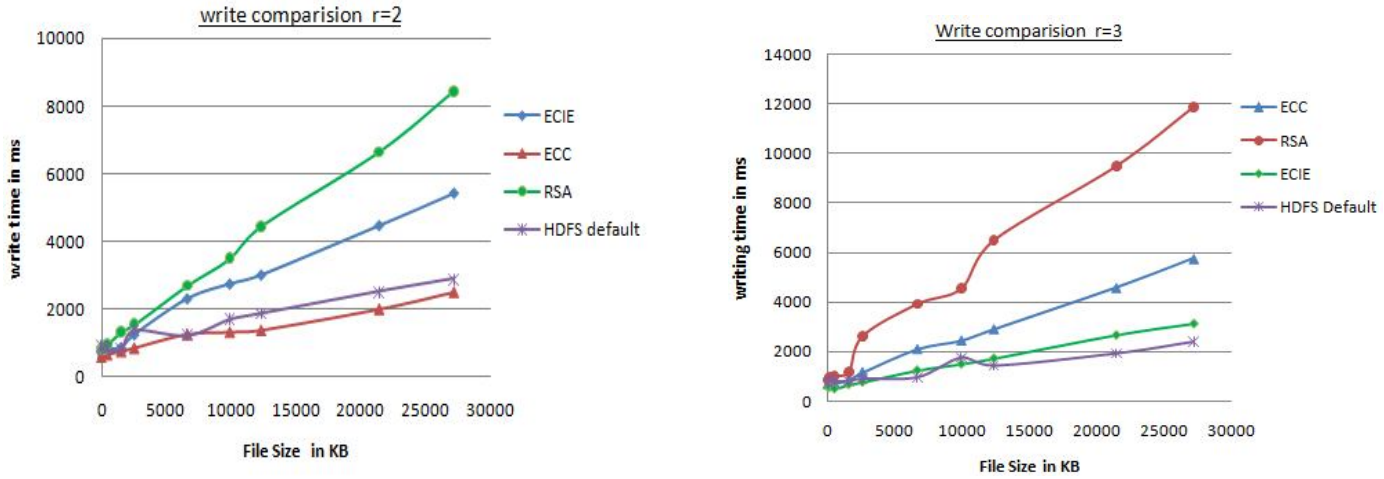

Fig 3. Write time comparisons in different schemes. Figure on left for scenario 1 and right figure for scenario 2

TABLE II
EXPERIMENTAL RESULTS OF READING FILE WITH DIFFERENT SCHEMES

Read time in ms								
File size in KB	RSA		ECC		ECIE		HDFS Default	
	rep=2	rep=3	rep=2	rep=3	rep=2	rep=3	rep=2	rep=3
2	625.80	634.00	467.90	491.10	525.00	463.33	825.50	685.30
94	630.70	671.00	445.50	495.80	500.90	460.22	872.75	655.60
463	741.20	745.33	438.60	515.80	549.70	480.89	1139.50	734.50
1537	969.10	1120.78	505.60	769.90	761.80	822.78	1235.00	734.90
2547	1410.30	1356.78	579.60	824.40	975.80	1033.44	1822.00	926.50
6630	2588.60	2075.11	788.20	1116.40	1428.50	1359.67	1209.00	880.70
9911	3485.80	2999.00	954.40	1521.70	1695.40	1777.22	1426.63	1140.60
12326	4241.50	4307.44	1127.30	1712.90	1597.00	2169.56	1821.00	1166.00
21428	6912.70	7620.22	1358.90	2427.10	3334.30	2879.44	2061.88	1305.70
27191	8340.00	9259.00	1922.30	3230.40	4239.90	3098.11	2776.38	1421.70

With replication factor set to 3 the performance of RSA is still poor of them all as depicted from graph. Here ECIE performs better than ECC and quite close to default HDFS.

D. Read performance analysis

The experimental results for reading times in each of four cases with replication factors 2 and 3 are shown in table II. As shown in figure 4 the again performance of the HDFS with RSA is worst while reading files when replication factor is 2. The overhead is directly proportional to size of the file. On the other hand ECC gives best performance which is followed by ECIE.In both the cases the read time tends get gradually stable with increasing file size. While reading the files the performance is hit by the overhead of decrypting the file contents in case of RSA, ECC and ECIE.

In case of RSA and ECC the reading overhead includes a) Reading the encrypted symmetric key b) Carrying out the decryption of the symmetric key c) Decryption of File contents. Even though the integrity check is performed by ECIE apart from the aforementioned three operations, it performs better than RSA. While in first two schemes two files have to be read from HDFS, the third scheme reads a single file as no separate file is stored. As shown in figure 4 on right, when the replication factor is set to 3 the performance of RSA degrades heavily with the increase in file size as in all the cases we discussed earlier. Here ECIE and ECC tend to approach same reading time and is getting changed little with very large file size.

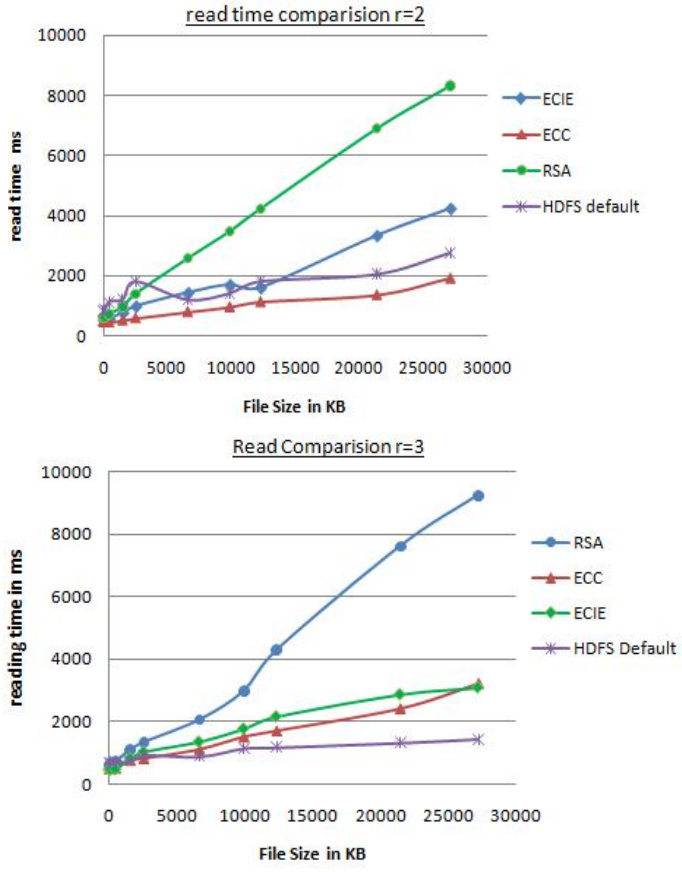

Fig 4. Read time comparisons in different schemes. Figure on left for scenario 1 and right figure for scenario 2

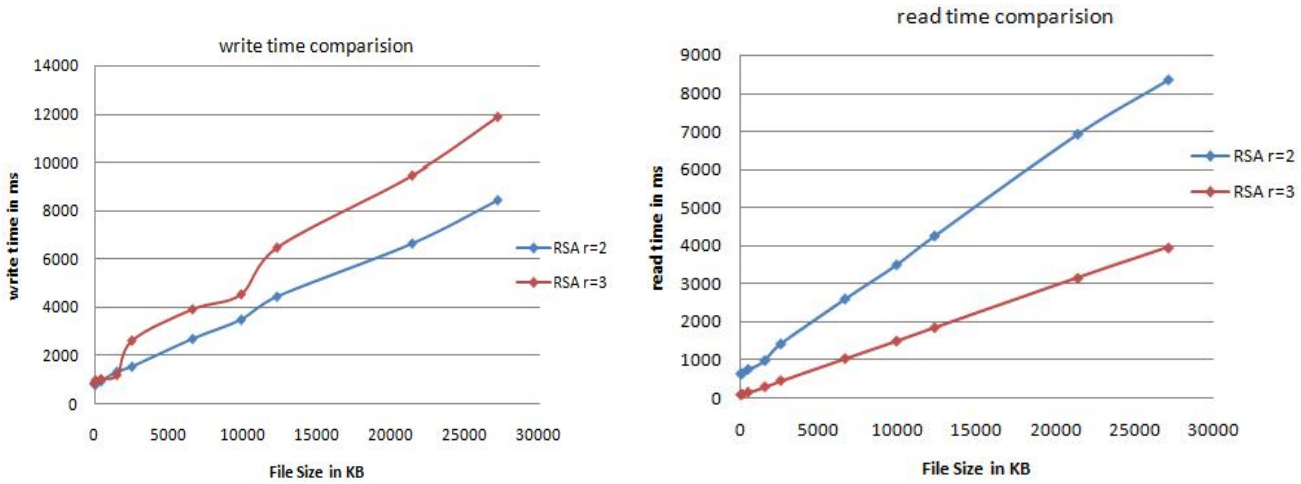

Fig 5: RSA performance comparison with replication 2 and 3. The left figure shows write time and right figure shows read time

E. Comparing performance of individual schemes with change in replication factor

When the analysis of time taken in read and writes operations is done in the case of HDFS with RSA with replications factors configured to 2 and 3, it can be observed
from the graphs in Figure 5 that there is not much difference in time taken while writing files of smaller size, however when the file size increases the more time is consumed. However in case of read operation the time taken is significantly less in case of replication value 3. According figure 6 the time to copy files as well as retrieving files from HDFS with ECC is less in case of replication value 2 but little higher with replication 3. It can be observed from figure 7 that writing and reading operation of files on HDFS with ECIE when the replication factor is 3 takes less time than with the two instances of files are stored. And the time tends to get stable in both read and write operations with replication factor 3.

F. Storage Overhead

As far as the client side storage overhead is concerned in our implementation is just the storage of Public and Private keys in all the schemes. The RSA and ECC public and private keys of user don't take more than 1 KB. We are storing the unique randomly generated asymmetrically encrypted secret key for each user file on server. In case of RSA it is just 128 Bytes and in case of ECC it is 73 bytes only. Whereas in case of ECIE no separate file is used to store anything ECIE inserts ephemeral Public key and tag at the beginning of the original file whereby modifying the original file size. However the extra bytes added in file are at most 52 bytes.

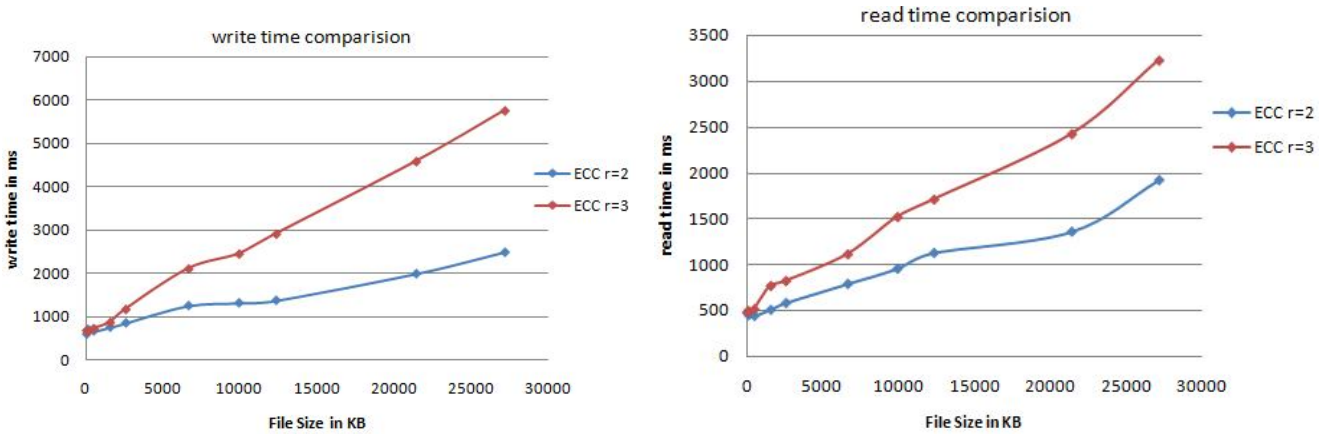

Fig 6. ECC performance comparison with replication 2 and 3. The left figure shows write time and right figure shows read time.

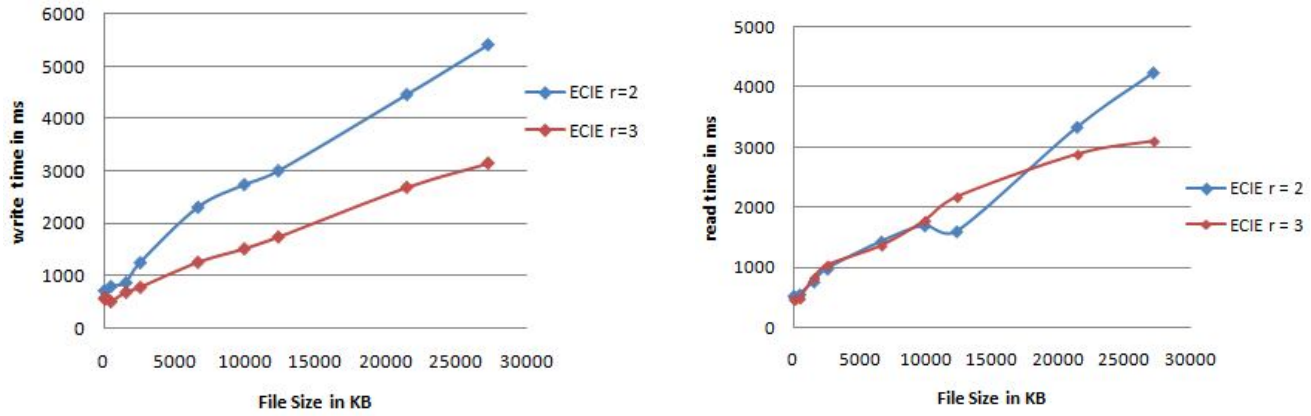

Fig 7 ECIE performance comparison with replication 2 and 3. The left figure shows write time and right figure shows read time

V. CONCLUSION

We have introduced an approach which is based on Elliptic Curve Integrated Encryption System to harness the Hadoop Distributed File System with security. In addition to provisioning for data confidentiality our implementation also provides integrity of user's data. Also a new random secret key is generated for each file that is stored on HDFS. However the user is freed from the overhead of secret key management as it is transparent to user. All user have to do is manage the Public and Private Keys of Public Key Encryption. The encryption and decryption of files when written to and read from Hadoop poses performance overhead. However for files of small size performance overhead is negligible so it is better suited for such applications. Our security integration in HDFS adds very small storage overhead on server and client.

VI. ACKNOWLEDGMENT

I thank Prof. B.S.Bhatt from Dharmsinh Desai University for his valuable guidance and support in this research.

REFERENCES

[1] Amit Sangroya, Saurabh Kumar, Jaideep Dhok, and Vasudeva Varma, "*Towards Analyzing Data Security Risks in Cloud Computing Environments* "

[2] Kevin Fogarty, "*The Biggest Cloud Computing Security Risk Is Impossible to Eliminate*", http://www.networkcomputing.com/security/the-biggest-cloud-computing-security-ris/240005337?pgno=2

[3] Tim Mather, Subra Kumaraswamy, Shahed Latif, "*Cloud Security and Privacy*"-O'Reilly

[4] Sashank Dara, "*Confidentiality without Encryption For Cloud Computational Privacy Chaffing and Winnowing in Computational-Infrastructure-as-Service*"

[5] K.Mukherjee and G.Sahoo "*A Secure Cloud Computing*", Recent Trends in Information, Telecommunication and Computing (ITC),International Conference on12-13,March 2010, Page(s): 369 - 371

[6] Stephen S. Yau and Ho G. An "*Protection of Users' Data Confidentiality in Cloud Computing*",2010 International Conference on Recent Trends in Information, Telecommunication and Computing, ISBN: 978-1-4503-0694-2

[7] Sushil Jajodia, Witold Litwin and Thomas Schwarz,"*Privacy of Data Outsourced to a Cloud for Selected Readers through Client-Side Encryption*",WPES'11, October 17, 2011, Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, pp 171-176, ISBN: 978-1-4503-1002-4

[8] Krishna P. N. Puttaswamy, Christopher Kruegel, Ben Y. Zhao, "*Silverline: Toward Data Confidentiality in Storage-Intensive Cloud Applications*"-SOCC'11, October 27–28, 2011

[9] Dai Yuefa, Wu Bo, Gu Yaqiang, Zhang Quan, Tang Chaojing, "*Data Security Model for Cloud Computing*", ISBN 978-952-5726-06-0 Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009) Qingdao, China, November 21-22, 2009

[10] Ethan Miller Darrell Long William Freeman and Benjamin Reed,"*Strong Security for Distributed File Systems*", 34 – 40, Performance, Computing, and Communications, 2001. IEEE International Conference Date of Conference: Apr 2001

[11] Fangyong Hou1, Dawu Gu2, Nong Xiao1, Yuhua Tang1, "*Secure Remote Storage through Authenticated Encryption, International Conference on Networking, Architecture, and Storage*", IEEE International Conference on Networking, Architecture, and Storage, ISBN-978-0-7695-3187-8,2008 IEEE

[12] Konstantin Shvachko, Hairong Kuang, Sanjay Radia and Robert Chansler,"*The Hadoop Distributed File System*", Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), p.1-10, May 03-07, 2010

[13] Qingi Shen,Dandan Wang and Min Long,"*SAPSC: Security Architecture of Private Storage Cloud Based on HDFS*", waina, pp.1292-1297, 2012 26th International Conference on Advanced Information Networking and Applications Workshops, 2012

[14] Joppe W. Bos, Marcelo E. Kaihara and Thorsten Kleinjung, "*On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography*", IACR Cryptology ePrint Archive 2009: 389, 2009

[15] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, Sheueling Chang Shantz "*Comparing Elliptic Curve ryptography and RSA on 8-bit CPUs*", 6th International Workshop Cambridge on Cryptographic Hardware and Embedded Systems, ISBN 3-540-22666-4, Aug 2004.

[16] Pavel Bzoch and Jiri Safarik "Security and Relaibility of Distributed File Systems", Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference,15-17 Sept.2011,Volume: 2 pp 764 - 769

[17] http://hadoop.apache.org/core/docs/current/hdfs_design.html

[18] Owen O'Malley, Kan Zhang, Sanjay Radia, Ram Marti,and Christopher Harrell. "*Hadoop security design*". https://issues.apache.org/jira/secure/attachment/12428537/ securitydesign.pdf, October 2009.

[19] Hsiao-Ying Lin, Shiuan-Tzuo Shen, Wen-Guey Tzeng, Bao-Shuh P. Lin, "*Towards Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System*",March 2012 26th IEEE International Conference on Advanced Information Networking and Applications, 978-1-4673-0714-7,p 740-741

[20] V. Gayoso Martínez, L. Hernández Encinas, C. Sánchez Ávila, "*A Survey of Elliptic Curve Integrated Encryption System*", Journal Of Computer Science And Engineering, Volume 2, Issue 2, AUGUST 2010

[21] Brian Warner,Zooko Wilcox-O'Hearn and Rob Kinninmont,"*Tahoe: A Secure Distributed Filesystem*",Mar 2008, https://tahoe-lafs.org/~warner/pycon-tahoe.html

[22] http://bigdata.wordpress.com/2010/03/22/security-in-hadoop-part-1/

[23] Samuel Sheinin,"*NFS Security*",Global Information Assurance Certification Paper, SANS Institute 2000 – 2002

[24] http://www.coda.cs.cmu.edu/doc/html/sec-1.html

[25] John.H.Howard,"*An overview of the Andrew File System*",CMT-ITC-88-062

[26] Tran Doan Thanh1, Subaji Mohan1, Eunmi Choi, SangBum Kim2 and Pilsung Kim,"*A Taxonomy and Survey on Distributed File Systems*", Fourth International Conference on Networked Computing and Advanced Information Management, 2008, 978-0-7695-3322-3/08