http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (12): 2398-2404, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.2398.2404

Improving Security of Parallel Algorithm Using Key Encryption Technique

[1]G. Sujitha, [2]M. Varadharajan, [1]Y. Vignesh Rao, [1]R. Sridev, [2]M.K. Sarvesh Gauthaum,
[2]Sowmy Narayanan, [1]R. Swarna Raja and [2]S. Mercy Shalinie
[1]Department of Information Science and Technology, College of Engineering,
ANNA University, Chennai, India
[2]Department of Computer Science, Thiagarajar College of Engineering,
ANNA University, Chennai, India

Abstract: In the recent cloud era, computing moves to a new plane of running large scale scientific applications. Many parallel algorithms have been created to support a large dataset. MapReduce is one such parallel data processing framework adopted widely for scientific research, machine learning and high end computing. The most prevalent implementation of MapReduce is the open source project Hadoop. To protect the integrity and confidentiality of data uploaded, MapReduce introduced a Kerberos-based model with tokens for datablocks and processing nodes. The tokens are symmetrically encrypted and distributed across the nodes. Such a technique is vulnerable to man-in-the-middle attacks like data loss, data modification and stealing of keys. In this study, a novel technique is proposed based on public key encryption on top of the Kerberos model to enhance Security. The various attack scenarios on the current Hadoop implementation model has been analyzed and a secure environment has been proposed. The study shows that the proposed framework provides an improved level of security when using RSA (Rivest Shamir Adleman) with 65,537 keysize consumed 23 milli seconds, while using 257 bits keysize which consumed 21 milli seconds.

Key words: Cloud security, mapreduce, dataintegrity, keymanagement, datasecurity, intrusion and attack detection

INTRODUCTION

Cloud computing being the peak of Gartner technology hype curve (Lapkin, 2012), proves that it matures and moves to a new plane of running large scale scientific and small scale problems. The need for large scale data processing (Ekanayake *et al*., 2008) has been reiterated in recent years due to the rising number of web applications with cheap storage capacity and availability of resources to process petascale of data. Several parallel algorithms (Isard *et al*., 2007; Mackey *et al*., 2008; Dean and Ghemawat, 2008) like Message Passing Interface (MPI), MapReduce, Dryad have been created with a view to efficiently run large scientific problems on a cluster of machines.

Google's MapReduce is a parallel programming model introduced to handle large scale data stored in datacenters and search applications (Dean and Ghemawat, 2008). It enables parallel processing of large data by partitioning data among a commodity of machines providing high performance (Mackey *et al*., 2008). It is capable of scaling and scheduling a cluster of machines, partitioning the data, determining which nodes will perform which tasks and handling messages between nodes. (Ekanayake *et al*., 2008) This frame work is versatile and allows new users to access the power of distributed computing by using distributed storage and avoiding the transmission step with an improved level of security.

One such implementation of MapReduce is Yahoo's Hadoop. Hadoop is a top level Apache Software Foundation project (Becherer, 2010). Hadoop's MapReduce is an open source software that fall under the infrastructure of distributed environment to perform large data analytics. Hadoop has been successfully used by many companies (Buyya *et al*., 2009; Fox, 2011) including America On Line (AOL), Amazon, Facebook, (Hadoop Applications, 2010) and New York Times for running their applications on clusters. For example, AOL uses MapReduce to study the behavioral pattern of their users and work accordingly.

In the early 2010 Yahoo! developers (O'Malley *et al*., 2010) attempted to improve the state of security. Many security loopholes like poor default SASL (Simple Authentication Security Layer) quality of protection, incomplete authentication and lack of data security that flows between the nodes were identified (Wei *et al*., 2009). Introduced a replication based verification scheme which is a decentralized scheme to run MapReduce securely. They also proposed a way to detect misbehavior of

Corresponding Author: G. Sujitha, Department of Information Science and Technology, College of Engineering, ANNA University, Chennai, India

malicious users. Major concerns like SASL and securing the channel are achieved through Kerberos based Authentication and through Remote Procedure Call via Secure Shell mechanism. Pluggable WebUI (User Interface) is resolved by SPENEGO (a mechanism to authenticate a remote server) that is supported by all Web Browsers. New goals were introduced which includes strong mutual authentication (Wei *et al*., 2009) of users and services that are transparent to end users. The wide distribution of symmetric keys is resolved through the proposed methodology. In this study, the three technical challenges identified were resolved by transferring the tokens securely between mappers and reducers.

MAPREDUCE ARCHITECTURE

MapReduce is a computing framework where nodes are distributed geographically. The framework is developed with two primary components namely HDFS and MapReduce engine (Hadoop Applications, 2010). Initial versions of Hadoop has been designed for a secure environment assuming that MapReduce and Hadoop Distributed FileSystem (HDFS) would be used by users of a cooperating group. The MapReduce dataprocessing model comprises of NameNode and DataNodes. The NameNode is responsible for scheduling the tasks, job management, fault tolerance and load balancing mechanism among the DataNodes. DataNodes are the workers that store and retrieve blocks based on the request from clients or NameNode and report back to the NameNode periodically with lists of blocks stored on it. A Jobtracker daemon receives the splits of a Job and assigns the splits to different nodes running on NameNode and oversees the overall execution of the MapReduce process. It assists in co-ordinating the jobs by maintaining the events and configuration details of completed jobs. A TaskTracker assigned, assists in managing the execution of individual tasks on the nodes of the cluster that sends a periodic heartbeat to the JobTracker to reveal that the TaskTracker is still alive and is ready to accept new tasks of MapReduce Process.

Tokens for security: The new Hadoop version 0.21 provides improved security performance by creation of tokens at different layers of MapReduce process. Token is a SecretKey created by the NameNode and Jobtracker and is shared between the clients, TaskTracker and DataNodes. NameNode creates DelegationToken and BlockAccessToken and JobTracker creates JobTokens to perform MapReduce job. Tokens like BlockAccess and JobToken are designed with timestamp inorder to varify the tokens authenticity and its validity.

Delegation tokens: DelegationToken generated by NameNode is created to enhance the security of Hadoop by allowing user authentication and pass credentials to all tasks of a job. It is generated to prevent flooding of authentication requests at the start of a job and is shared as a secret key between the client and the NameNode/JobTracker. For subsequent calls, DelegationToken alone can be verified by NameNode for each user instead of using kerberos tickets.

BlockAccessToken: Inorder to securely access the contents of HDFS BlockAccessToken is created. Actually NameNode creates such tokens and is accessed by DataNode. To access a file, clients communicate with the NameNode to find out which DataNodes the user intends to access inorder to fetch the file. DataNodes need to know from the NameNode whether the client is authorized to fetch that blocks. Inorder to authorize the client, information about the client block_id is passed using BlockAccessToken from NameNode to the owner of the file. Using the token, the owner/client can access the data block.

JobToken: The token is created to securely run MapReduce jobs. It is generated by JobTracker and distributed to all MapReduce Tasks (TaskTracker Nodes) in order to run jobs by providing a check as whoever comes with the token is authentic to run. When task communicates with TaskTracker for results or computational purposes, JobToken is used. JobTracker automatically renews tokens while job is running and cancels tokens when job is finished and hence it is not persistent.

Flow of tokens: All communication between the nodes of distributed environment takes place via SecureShell (SSH). Despite transferring tokens via SecureShell, Hadoop users who access (Malley, 2010) Hadoop services are authenticated through a Kerberos authentication protocol. Hadoop Clients access its services via Hadoops Remote Procedure Call (RPC) library. Each users login name sent across the connection setup is authenticated through Kerberos based authentication to NameNode in the existing Hadoop 0.21 version as shown on the Fig 1. Kerberos has a key distribution center (KDC) which maintains AuthenticationServer and Ticket Granting Server. Authenticated client receives a timestamped (TicketGrantingTicket) TGT from authentication server. Ticket Granting Ticket is used to request a service ticket from ticket granting server. Client uses the service ticket to accept the delegation token from NameNode through

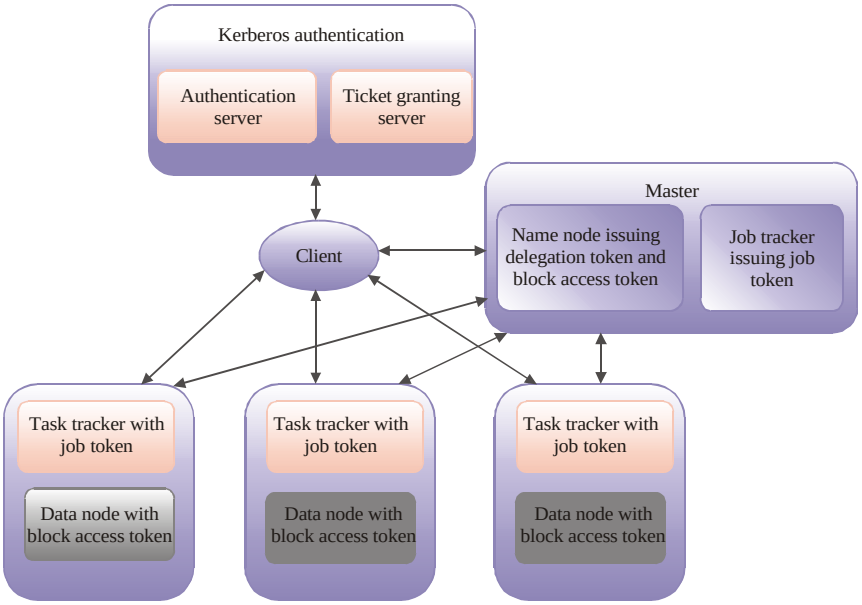

Fig. 1: MapReduce Architecture describing flow of tokens

RPC call. The three step Kerberos authentication for each call overloads the KDC on a cluster for which a DelegationToken is introduced.

Figure 1 describes how the Kerberos service ticket and tokens flow during a MapReduce process. After initial user authentication through Kerberos the whole process involves running MapReduce jobs on a Hadoop cluster and accessing HDFS blocks. To access HDFS using BlockAccessToken, the access information of a user is passed from NameNode to datanode. So BlockAccessToken issued by NameNode is used on datanode to verify its authenticity. The token enables its owner to access certain DataBlocks. Here DataNodes do not enforce access control on data blocks. This allows an unauthorized client to read a datablock when it knows the block_id. So insecure channels in between the nodes allow malicious users/attackers to fetch others blocks. MapReduce jobs running on a cluster accepts the jobtokens created, while running the task. The authenticity is checked with the users Delegation and JobToken.

Risks associated with handling tokens: Client Server interactions in a distributed environment like HDFS or MapReduce is prone to numerous attacks. So such an environment needs to be authenticated. Tokens are created and assumed to follow a secure channel. Besides communication security threats such as Denial of Service attacks, eavesdropping attacks, replay attacks, MapReduce faces issues while maintaining the integrity and confidentiality of data (Malley, 2010). Authenticated users obtaining a delegation token shares the token between the user and the NameNode. The delegation token need to be protected when passed over insecure channels.

An Attacker can affect the integrity of the MapReduce process in two ways. Firstly when a client asks NameNode for block_ids (locations) of a file on HDFS, NameNode checks that the client is authorized to access the file and sends back blockids along with a BlockAccessToken for each block. In the current version of Hadoop 0.21, BlockAccessTokens are generated using a symmetric HMAC-SHA1. It is distributed to NameNode and all DataNodes in the cluster. So, an attacker who fetches a shared key is vulnerable to attack all DataNodes as shown in the Fig. 2. Also the confidentiality of data is affected because of the attack. Secondly, while running jobs on a HDFS Cluster an intruder who is capable of acquiring JobTokens can modify the results of a Map or Reduce task.

Proposed technique to secure MapReduce environment: Several risks related to HDFS data integrity, privacy and confidentiality of Hadoop users needs to be reduced by performing cryptographic encryption technique on tokens when traversed from one node to another in a Hadoop environment. The three methodologies identified in cryptography prove to be secure in the proposed implementation.

Using asymmetric encryption mechanism: The proposed technique is to replace the symmetric HMAC-SHA1

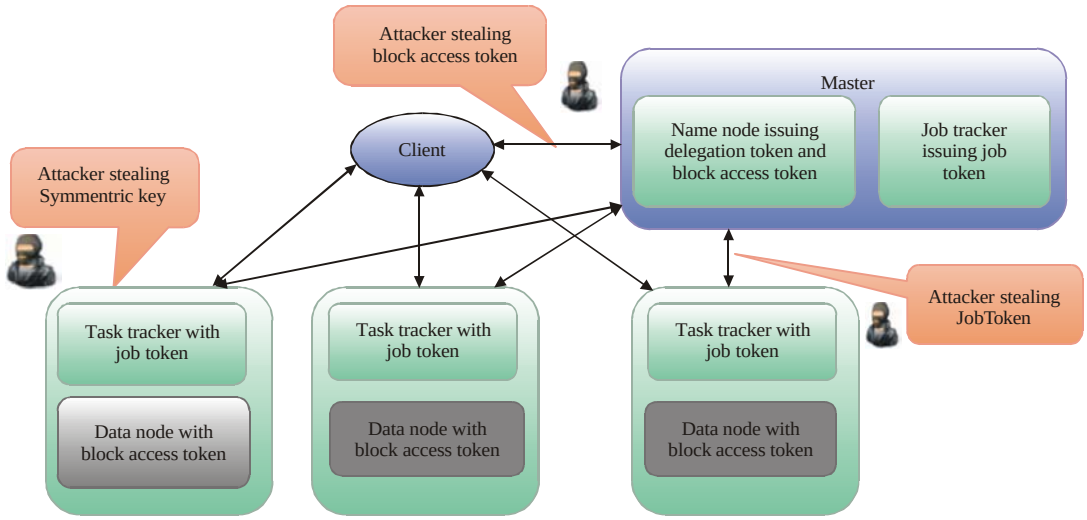

Fig. 2: Depicting an attacker stealing BlockAccessToken, SymmetricKey and JobToken when traversed between NameNode and DataNodes

system supported in the version Hadoop 0.21 by a public-key system. In the case of BlockAccessToken, instead of storing the same symmetric key over several DataNodes, a public-private key pair is generated between the NameNode and every datanode. Similarly, job tokens are encrypted using a public key shared by the JobTracker and the task. When a job is submitted by a Hadoop client, the JobTracker first generates key pairs with each of the TaskTracker. When a job is split to the various TaskTrackers, the job token ensures a secure communication. In this case, the token is encrypted by the key which is available only at the encrypting end i.e JobTracker and sent to the corresponding TaskTracker node. At this end, the TaskTracker decrypts the token using its private key and checks the authenticity. Inorder to ensure that token is arrived from the appropriate JobTracker, its digital signature is appended with the token. Similarly, during data access from HDFS, public keys for encryption are stored in the NameNode and each of the DataNodes have the corresponding private keys. The blocks at the respective DataNodes can be retrieved by verifying the corresponding encrypted tokens.

Figure 3 gives the algorithm which was implemented with methods like keyinit in line number (3) written for setting the default parameters like length of the key and type of algorithm which is RSA. The generateKeys function of crypto library in line 8 is for generating public and private keys. The private keys generated were based on different values of commonly used public key values such as 17, 257 and 65537.The public and private key pair created in the NameNode of cluster is shown in Table 1.

```
1  private static final String DEFAULT_ALGORITHM = "RSA";
2  private final KeyGenerator keyGen;
3          protected keyinit()
4          {
5            keyGen = KeyGenerator.getInstance(DEFAULT_ALGORITHM);
6            keyGen.init(KEY_LENGTH);
7          }
8  protected KeyPair generateKeys() {
9     KeyPair kp;
10    synchronized (keyGen) {
11          kp = keyPairGenerator.genKeyPair();
12          }return kp; }
13  protected static Keypair createPublicKey(byte[] key) {
14          return new KeyPair.PublicKey(key, DEFAULT_ALGORITHM);
15          }
16 protected static Keypair createPrivateKey(byte[] key) {
17          return new KeyPair.PrivateKey(key, DEFAULT_ALGORITHM);
18  }
19 }
```

Fig. 3: Proposed asymmetric algorithm using RSA for securing tokens in Hadoop Version 0 .21

Table 1: Asymmetric Keys created in NameNode and distributed to nodes of cluster

Nodes of cluster scheduled to run Map/Reduce Tasks	Private key	Public key
Node 31	4.59E+029	65537
Node 8	6.14E+017	17
Node 23	271041713	65537
Node 32	250889438643593	17
Node 17	5.70E+017	65537
Node 26	1.54E+020	65537

Using symmetric encryption mechanism: The second technique to replace the HMAC-SHA1 system is providing a symmetric-key encryption mechanism. The symmetric hashing is replaced by a symmetric encryption mechanism. In the case of BlockAccessToken, instead of storing the same symmetric key over several DataNodes a private key is generated between the NameNode and

every DataNode. Similarly job tokens are encrypted using a private key and is shared by the Jobtracker and each of the tasks. When a job is submitted by the user, the Jobtracker generates private keys and shares it with each of the TaskTrackers. When a job is split to the various TaskTrackers, the job token ensures the secure communication. The Jobtracker encrypts the Jobtoken by the private key. It is decrypted using the same private key and checked accordingly at the TaskTracker node. Inorder to access data from HDFS, BlockAccessTokens are encrypted using private keys that are maintained by NameNode and each of the DataNodes which receives the encrypted form of blockid from the user, checks the token by decrypting it with the same privatekey sent by the NameNode. The implementation of the symmetric algorithm is similar to that of the previous method except that now a single secret key is generated in place of the public and private key pair.

Refinement in the existing HMAC-SHA1 mechanism of Hadoop's MapReduceVersion 0.21: The third technique is based on the existing Hadoop Security Design (O'Malley *et al*., 2010). The model shares the same secret key across nodes to compute the hash value of tokens like BlockAccess and JobToken. The algorithm designed generates a unique secret key for each and every node instead of using the same key. Such concept introduced, reduces the risk of stealing or destroying the contents of all blocks of HDFS. Even a malicious user who steals a secret key is unable to access the contents from other blocks and is exposed only to the block for which the secret key is assigned. The implementation of such mechanism involves keyinit method in line4 to set the default parameters for the key. But generateKeys method in this case uses the actual crypto method to generate as many secret keys as the mapper nodes. Such technique shows relatively better results which is discussed in our experimental studies.

RESULTS

The algorithms proposed above were tested on a campus environment that is created as a TestBed. The TestBed comprises of a 32 node cluster, each of which has an Intel Xeon 1.6Ghz processor with 500 GB of local storage running on Hadoop 0.21.The initial tests were made with earlier versions of Hadoop .20, Hadoop .20 (secure version).The TestBed was tested with series of applications from wordcount, grep, sort for the key encryption techniques. The study revealed that the key encryption mechanism shows relatively less time when using Asymmetric Rivest Shamir Adleman (RSA) encryption technique than the symmetric Advanced Encryption Standard (AES) encryption. To prove that, experiments were explained with the time lapsed for keygeneration, keydistribution and encryption technique together and the comparison graph was generated. The experimental results show that the proposed framework provides an improved level of security where in RSA (Rivest Shamir Adleman) consumed 23 milli seconds (for 65,537 keysize) in comparison to the 257 bits keysize which consumed 21 milli seconds.

Experiment 1: Asymmetric encryption of tokens: For asymmetric encryption the TestBed is used initially for keygeneration on NameNode for fixed common values for public keys like 17,257 and 65537 and a series of values for private keys which varies based on the size of the keys as shown in the Table 2.With the public and private keys as generated in the table, tokens like BlockAccess and JobToken are transferred securely using encryption techniques. Such authentic users are alone allowed to access HDFS blocks and run MapReduce jobs.

Key generation for asymmetric encryption using publickey values based on their bit size for different bit values of private keys is shown in the Table 2. The Fig. 4

Table 2: Generation of public and private keys for varied key sizes

	Private key size in bits			
Public key based on bit size	8	16	32	64
17	53	31729	670549409	5.27e+018
257	93	26393	707865473	6.42e+018
65537	113	7973	1148897573	1.05e+018

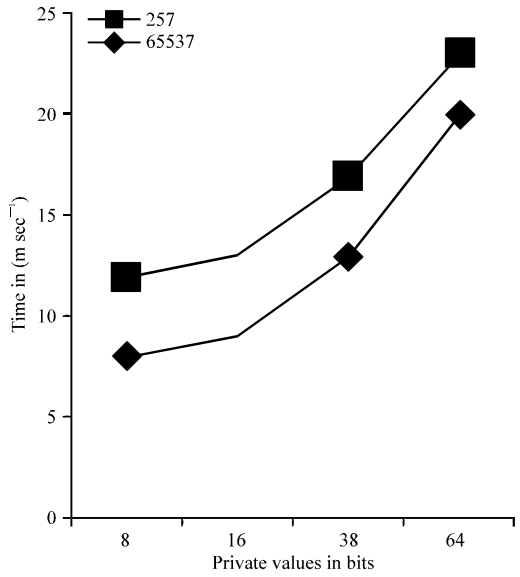

Fig. 4: Shows the time taken for key generation using asymmetric RSA using different public keys

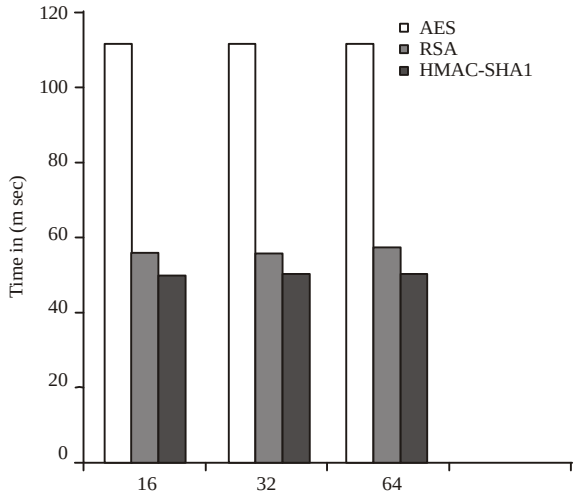

Fig. 5: Comparison of the cryptographic algorithm with the time taken vs different key sizes

shown below depicts that key generation using Asymmetric RSA using different public keys consumed relatively less time (23 ms) for its size, than choosing the key size with 257 bits which consumed 21 m sec. The results of Table 2 proved that public key with 65537 keysize was fixed to generate private keys.

Experiment 2: Comparison of the three techniques: The Fig. 5 shown is related with the time taken to generate the key, based on the number of nodes of a cluster, distribution of key and the overall time taken to perform encryption on 32 node TestBed. It was clear from the result that the study depicts that there is less overhead in creating the keys. The overhead incurred is because the keys traverse from NameNode to all the other nodes. As discussed, a SecureShell communication is implemented for the whole environment. The possibility of data loss in such an environment is highly impossible as the intruders need to crack the private key for each and every node of the cluster.

DISCUSSION

In this study, a comparison of Becherer (2010) Hadoop over Tahoe- Least Authority File System (LAFS) revealed that individual nodes of cluster, encrypt files on disk. It was designed to communicate over a secure transport in the case of Tahoe FileSystem (Wilcox-O'Hearn and Warner, 2008) and highlighted that Hadoop's Distributed File System (HDFS) does not securely maintain data. All DataNodes share a secret key (Armbrust *et al*., 2010; 2009) when accessing the data from HDFS. Such a mechanism is vulnerable to attack in case of running the model in an insecure channel. The issues related to privacy, integrity and authenticity identified from the papers (Becherer, 2010, Wei *et al*., 2009) motivates to perform a key encryption technique to enhance Hadoop Security. The various issues of Hadoop's MapReduce like insufficient authentication along with factors affecting the privacy and integrity of data stored on HDFS were projected.

The experimental results prove that usage of asymmetric methodology involves more overhead for an attacker to fetch the data from the tasktracker or datanode when compared to the other two techniques like symmetric and unique hash function mechanism. The experiment is not tested for various applications of MapReduce like sort and grep since the performance of the model designed is based on the type of cryptographic mechanism used like symmetric or asymmetric method. The experimental result of the proposed technique shows that key encryption technique requires less time in the order of milli seconds for a 32 node cluster. When the nodes are geographically distributed and large dataset submitted to nearly 1000 nodes of cluster key computation and distribution complexity can be reduced by configuring multiple NameNodes of Hadoop Cluster. The overall performance of the above experiments reveals that RSA a asymmetric cryptosystem proves to be a strong algorithm for securing Hadoop Environment.

CONCLUSION AND FUTURE ENHANCEMENTS

A secure MapReduce model is proposed and implemented that is adaptable for running any application. The goal is derived with a view to provide an improved level of security to MapReduce framework that is being used widely in Cloud in recent years. The model is designed with providing three levels of security first at user level, second at the MapReduce process level and third at the HDFS level. The first level is achieved using Kerberos authentication followed by running MapReduce securely using tokens that are encrypted when they traverse over wire. The HDFS access is done securely through encryption technique on tokens which access the blocks. The authentication for the user using kerberos is followed by authentication at namenode level, tasktracker level while running the task and checking for authentication by datanode when accessing a HDFS block for users. HMAC-1 supported by Hadoop v.21 is replaced with public key mechanism using RSA algorithm. Such policies enforced in the new model provide high level of security for Hadoop's MapReduce and HDFS. To reduce the complexity involved in maintaining the tokens,

a simple hashing technique methodology is explained that offers a reliable way to run the MapReduce process. Such mechanisms prevent Distributed Denial of Service attacks, Replay attacks on nodes involved in MapReduce process and stealing of keys from the nodes of cluster. The experimental result of the proposed technique shows that key encryption technique achieved relatively less time 23 milli seconds even when the number of nodes of the cluster is high. Through open source innovation the above system offers a secure model for an increased impact on Hadoop community all over the world. The scope of the project is aimed to secure HDFS blocks further by performing an effective encryption mechanism on the datablocks thereby serving cloud store in the recent Cloud Era.

ACKNOWLEDGMENT

We acknowledge with thanks, Yahoo! India, (Banglore) for providing research grant to support this study.

REFERENCES

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph and R.H. Katz *et al*., 2009. Above the clouds: A Berkeley view of cloud. Technical Report No. UCB/EECS-2009-8. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph and R. Katz *et al*., 2010. A view of cloud computing. Commun. ACM., 53: 50-58.

Becherer, A., 2010. Hadoop security design, just add kerberos? really?. iSEC Partners, Inc.

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 25: 599-616.

Dean, J. and S. Ghemawat, 2008. Mapreduce: Simplified data processing on large clusters. Comm. ACM, 51: 107-113.

Ekanayake, J., S. Pallickara and G. Fox, 2008. MapReduce for data intensive scientific analyses. Proceedings of the IEEE 4th International Conference on eScience, December 7-12, 2008, Indianapolis, IN., USA., pp: 277-284.

Fox, A., 2011. Cloud computing-what's in it for me as a scientist? Science, 331: 406-407.

Hadoop Applications, 2010. Powered by Hadoop Wiki. http://wiki.apache.org/hadoop/PoweredBy.

Isard, M., M. Budiu, Y. Yu, A. Birrell and D. Fetterly, 2007. Dryad: Distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operat. Syst. Rev., 41: 59-72.

Lapkin, A., 2012. Hype cycle for big data. Gartner Inc. http://www.hadoopconsultant.nl/wp-content/uploads/hype_cycle_for_big_data_2012_235042.pdf

Mackey, G., S. Sehrish, J. Lopez, J. Bent, S. Habib and J. Wang, 2008. Introducing mapreduce to high end computing. Proceedings of the 3rd Petascale Data Storage Workshop, November 17, 2008, Austin, TX., USA.

Malley, O., 2010. Integrating kerberos into apache hadoop. Kerberos Conference 2010, MIT, Cambridge, UK.

O'Malley, O., K. Zhang, S. Radia, R. Marti and C. Harrell, 2010. Hadoop security design. http://carfield.com.hk/document/distributed/hadoop-security-design.pdf

Wei, W., J. Du, T. Yu and X. Gu, 2009. SecureMR: A service integrity assurance framework for mapreduce. Proceedings of the Annual Computer Security Applications Conference, December 7-11, 2009, Hawaii, USA., pp: 73-82.

Wilcox-O'Hearn, Z. and B. Warner, 2008. Tahoe: The least-authority filesystem. Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, October 31, 2008, Virginia, USA.