# Big Secure Data Storage a Hybrid Encryption Mechanism For Hadoop Environment

Amit Gupta[1], D. Rajani[2], R Jawwharlal[3] , Arun Singh Chouhan[4]
[1]Associate Professor , Department of ECE , Nalla Malla ReddyEngineering College,email-id :dramitguptacv@gmail.com
[2]Associate Professor , Department of ECE , Nalla Malla ReddyEngineering College,email-id:rajani.ece@nmrec.edu.in
[3]Associate Professor, Department of ECE ,Nalla Malla ReddyEngineering College,email-idrjlphdou@gmail.com[4]Associate Professor, School of Engineering, Malla Reddy University,email-id:arunsingh.chouhan@gmail.com

**Abstract—**Hadoop is an open-source framework that use the map reduce architecture to store and analyse large datasets concurrently. As both inside and outside attacks may jeopardise a network's security, keeping this data safe is of the utmost significance. Although Hadoop does have a number of different authentication methods, only it takes into consideration a subset of possible security solutions. For this reason, we propose a technique that evaluates the efficiency of the encrypted multi-node hadoop cluster. As far as can be told, data may be stored in a file system called Hadoop Distributed File System (HDFS) that is shared across a group of computers. Our technology allows for the RSA-SHA encryption of any file or directory tree, making it a data-centric security solution. A K-means clustering mapreduce technique is used for streamlined distributed processing of massive volumes of files on clusters. Overall, our technology strengthens data protection while also facilitating simple and rapid access to previously stored files.

***Keywords*—Secure Hadoop Cluster, Hybrid Encryption, HDFS, and RSA-SHA Algorithm**

## I. INTRODUCTION

A free framework called Hadoop can efficiently store a lot of data in a cluster. It is based on Java. We may process data across all nodes using this framework's ability to operate in parallel on a cluster. Hadoop settings contain data from several sources with varying degrees of security sensitivity, including weblogs, online transactions, social media interactions, etc., necessitating the need for security. Hadoop's initial architecture didn't prioritize security.The huge volumes of digital information that are gathered, combined, and analyzed to produce "big data" are what give it its worth. Its goal is to increase the volume of data that can be processed and stored. Big data is characterized by its diversity and complexity, the former of which relates to the multitude of data sources that must be analyzed and the latter to the always increasing amount of data flows.

Following are three characteristics of big data: volume, diversity and velocity.

**Volume**: Large datasets, or "big data," can range in size from a few gigabytes to a terabyte (1,000 terabytes). It demands the support of various data sources and the scalable storage of complex distributed queries.

**Velocity**: Making judgments using big data, which processes both real-time and historical data in parallel, enables more precise and quick conclusions.

**Variety:** The big data framework supports data from a wide range of sources, including multimedia, RFID tracking devices, web server logs, financial transactions, social media, GPS, etc.

With so many innovations enabling the growth of data, technology has advanced dramatically in recent years. The most prevalent illustration of a large amount of data is the internet search history. Many individuals all throughout the world will look for the knowledge they need. Managing this enormous amount of data becomes a very difficult problem. The idea of Hadoop was created in order to process these data. Hadoop will make it possible to store and process data in parallel while storing it in a distributed fashion. A master-slave architecture underpins the Hadoop cluster storage system.

The development and implementation of programmers to handle massive data collections might be sped up using the Hadoop platform. Hadoop is free and open-source software that was initially created to accelerate the process of searching for files over a network. It is distributed under the Apache Open-sourced License version 2.0. Hadoop was created to handle the numerous issues that occur when working with enormous volumes of data in addition to providing an effective big data infrastructure. Hadoop's primary building blocks may be classified into two categories:

**I.Processing (Map Reduce)**

Using a distributed, synchronous technique on a cluster, lossless compression is in fact a software design idea and its supporting implementation for processing and producing big data sets. The Reduce procedure [method] in a Map Reduce programmer equips an immediate procedure [including such as tallying the students present in each queue, yielding name frequencies], in contrast to the Map procedure, which is responsible for sorting [including such things as trying to sort educators by first name into queues, one queue for each name] and filtering. The Map Reduce System, a structure or framework in charge of coordinating the system's distributed servers, attempting to carry out its numerable tasks in parallel, overseeing its information flow among its many components, and providing for redundancy, can be credited with the high processing scores.

### II Map Reduces Method

Although its implementations inside the Map Reduce architecture are significantly different from their forebears, these paradigms are based on the map and reduce techniques that are often used in functional programming. The main determinants of the Map Reduce structure is not the map and reduce functions themselves [which, for example, mimic the decrease and disperse operations of a Common User Interface standard]; rather, it is the part of this exercise & strain acceptance achieved for a wide range of applications in addition to improving the execution engine once. Consequently, it is uncommon for a Map Reduce implementation to outperform a non-Map Reduced counterpart, and multi-threaded solutions are typically required to get any speed improvements.
The idea won't be taken seriously until the better distributed shuffle operation and fault tolerance of the Map and Reduce application are implemented. How effectively a Map Reduce algorithm can reduce communication costs has a significant impact on how successful it is.The level of optimization included in different languages' Map And Reduce libraries varies. Apache Hadoop is a well-known open-source technology that can do distributed shuffles. Although "Map Reduce" was previously solely used to describe a Google-exclusive technique, it is now frequently used. The development of Apache Mahout has migrated to devices that are both more competent and less disk-oriented, including complete map and reduce capabilities. Map Reduce is no longer Google's core Big Data processing architecture. The map phase of Hadoop, which is a component of the map reduction paradigm and may be utilised to quickly retrieve data in a cloud setting, is driven by our suggested approach's jetty server. MapReduce is a programming paradigm that combines the Map and Reduce functions. To make handling the data easier, it has been divided into manageable segments. These data blocks are then duplicated by HDFS and sent to the cluster's other nodes. Finally, Map Reduce is capable of processing data straight from the original storage places.The map phase creates the intermediate sets by taking a pair of input keys and a set of output keys.Phase of reduction in which the input key-value pair is combined with another key-value pair and the result is output.

## II.Literature Reviews

Information and knowledge extraction heavily relies on computing infrastructure, particularly Cloud computing. Traditional data encryption standards, techniques, and algorithms frequently face new security concerns as a result of the efficient management of huge data. Due to performance and scalability difficulties, previous studies of data encryption tended to focus on small-to-medium-sized data, which does not work well for big data.

As a result, efficient data access control and safety management rules must be developed for handling huge data, which necessitates the use of new data management systems.As an open-source big data and cloud computing framework, Hadoop is being utilized more and more in the corporate sector, however Jam et al. reported[1] that one of the primary issues impeding its growth is the lack of adequate security measures. This article begins by outlining the Hadoop project and its current security measures. It then analyses the security issues and threats it poses, considers possible ways to improve its trust and security, and, based on the foregoing descriptions, draws a conclusion on Hadoop's security difficulties.

D. Shehza et al. researched a unique hybrid encryption technique that combines an asymmetric data key algorithm with RSA and a symmetric key approach employing pictures as secret keys. Comparing the proposed system to the other encryption methods already in use, the overhead of the secret key computation cycles was minimized.In this study, a unique hybrid encryption method that combines the Advanced Encryption Standard (AES) and Cipher-text Policy Attribute-Based Encryption (CP-ABE) algorithms is used to protect massive data. To demonstrate the suggested model's improved performance in terms of throughput, encryption time, decryption time, and efficiency, it is compared to well-known encryption algorithms like DES, 3DES, and Blowfish. The suggested model outperforms with a maximum efficiency of 96.5% with 7.12 min of encryption time and 6.51 min of decryption time.

Secure large data utilizing a unique hybrid encryption technique that combines Cipher-text Policy Attribute-Based Encryption (CP-ABE) and Advanced Encryption Standard (AES) algorithms, according to T. Mohanraj et al. [3]. To demonstrate the suggested model's improved performance in terms of throughput, encryption time, decryption time, and efficiency, it is compared to well-known encryption algorithms like DES, 3DES, and Blowfish. The suggested model surpasses traditional encryption methods with a maximum efficiency of 96.5% with encryption and decryption times of 7.12 and 6.51 minutes, respectively. adopt safety Hadoop was developed in 2008 with the goal of managing only massive amounts of data that were restricted to a certain situation. Security concerns thus weren't the primary priority[4].

The user's name is used by Hadoop for all data storage. There is no encryption between Hadoop, the client host, and the HDFS in the default node. A centralized server called NameNode receives all the records and controls them. preserving shared calculations that are secret using mapreduce. Loss of data control, which poses a severe threat to client confidentiality when adopting cloud computing, is the primary source of cloud vulnerability. By offering safe computer environments and data storage, this issue may be solved[5].

For the safe storage of files in the cloud, Vinay Prodigal et al. [6] suggest using hybrid cryptography and picture steganography methods. Due to the employment of various algorithms for the encryption/decryption process, this technology aids in attaining improved efficiency and better security. In an effort to improve findings and increase the security of the sent data, this study has added work utilizing the 3-DES method for encryption. In the financial and private sectors, where the suggested technology can be deployed, high level data protection is necessary.

## III.PROPOSED MODEL

Non-repudiation is the most important criteria from the point of view of security. Digital signatures are an essential security basic because they provide the non-repudiation feature. A digital signature is characterized as a cryptographic operation that upholds non-repudiation, authentication, and permission. Typically, the authentication procedure makes use of digital signatures. One will require their own private and public keys in order to establish a digital signature. Symmetric key encryption techniques are therefore inappropriate for creating digital signatures. Mathematical formulas are used to connect the two keys and produce a digital signature. The majority of digital certificates now authenticate users using digital signatures.

### *A.* Digital Signature Generation

The production of digital signatures typically occurs on the sender side. Both message digest generation and digital signature creation are steps in the process of creating a digital signature. The inability of the signature to be duplicated and reused is guaranteed by the digital signature. This is so that the signature won't match with other material as it was created using a specific document's content. Creating a digital signature involves the two stages listed below.

### *B.* Sign Generation

Before being signed, data is hashed to ensure that each piece of data can be uniquely identifiable. A hash value is a brief and distinctive identifier, similar to a fingerprint. Hash functions are frequently used in almost all other information security procedures. In reality, creating hashes is a mathematical procedure for compressing data. Any length of input data may be used in hashing to generate hashes of a certain length. Once calculated, these values are referred to as messages digests or hash codes. The possibility that a hash may, among other things although hash functions are frequently simple to build, it may be computationally expensive to reverse the hashing process in order to obtain the original input using only the hash value. Due to this fact, an attacker will be impossible to recreate the original data even if they know the hash value.

• Although hash functions are frequently simple to build, it may be computationally expensive to reverse the hashing process in order to obtain the original input using only the hash value. Due to this fact, an attacker will be impossible to recreate the original data even if they know the hash value.

• The hash value is constructed to ensure that no two input values will provide the same hash value. Each input value is separately hashed. In order to locate an alternative input by merely altering the inputs and their hash value, an adversary would need to be aware of both the original input and the hash value.

• As a result of the hashing method's assurance of a collision-free property, it would be exceedingly difficult for an attacker to find two input values. The Figure 1 and 2 illustrated that digital signature generation and figure 2 Digital signature verification.

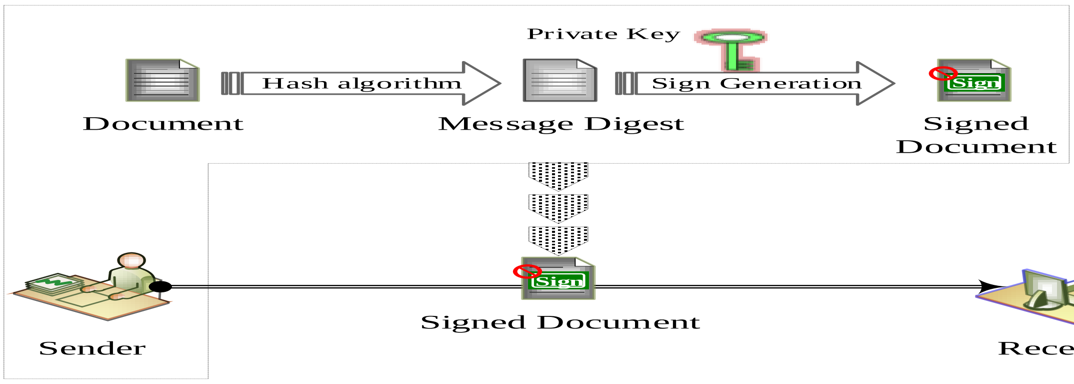


Fig.1. Digital signature generation

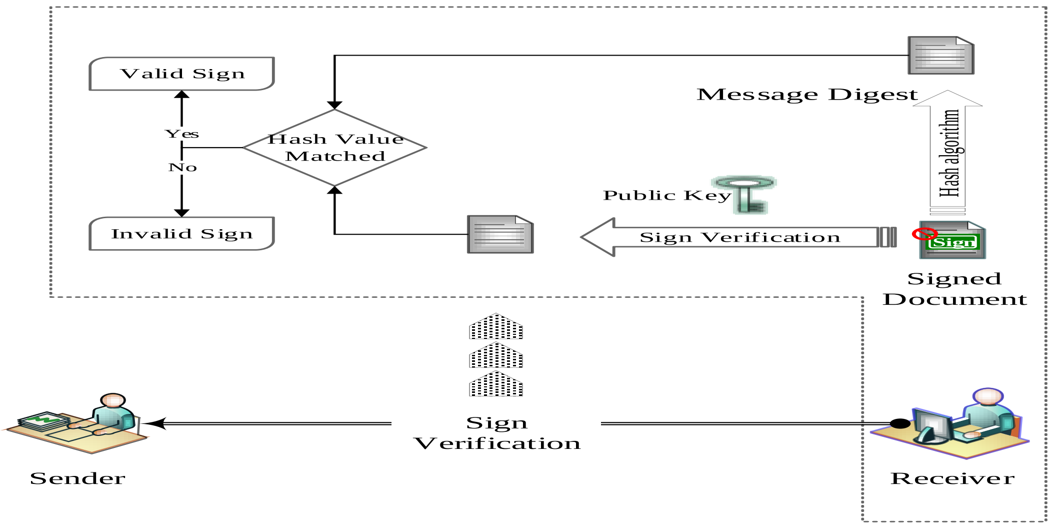


Fig.2. Digital signature verification

The sender then uses the private key to sign the hash value. As a result, anybody else cannot create a digital signature without the original private key. The produced digital signature is then added as an appendix to the data or document for further security.Figure 2 illustrates the steps involved in creating a digital signature. The procedure begins with the signer creating a hash value for certain data from the document. The data is then included with the signature. Finally, the recipient receives the data that has been digitally signed.

## C. Digital Signature Verification

The verification of the signature demonstrates the reliability of the signature. The receiver and authentication administrator might both be the verifier. The verifier initially creates a hash value for the data it has received.

After that, the signature is verified using the sender's public key. As a signature process, this procedure is reversed. The hash value, which the sender had previously attached, is gleaned from the verification procedure. Next, a comparison is made between the hash value produced by the verifier and determined from the data. The signature is valid if the hash value matches. The signature is not valid if anything else. Figure.3shows the procedure of signature verification. It is important to note that the signature verification checks the data's integrity as well. The hash value will be different if the attacker modifies or tampers with the data while it is being sent. Therefore, utilising a digital signature raises the bar for security. The suggested solution incorporates the encryption and digital signature processes to increase the security of files stored in the Hadoop Distribution File solution (HDFS).The figure 3 illustrated that flow diagram of signature data.

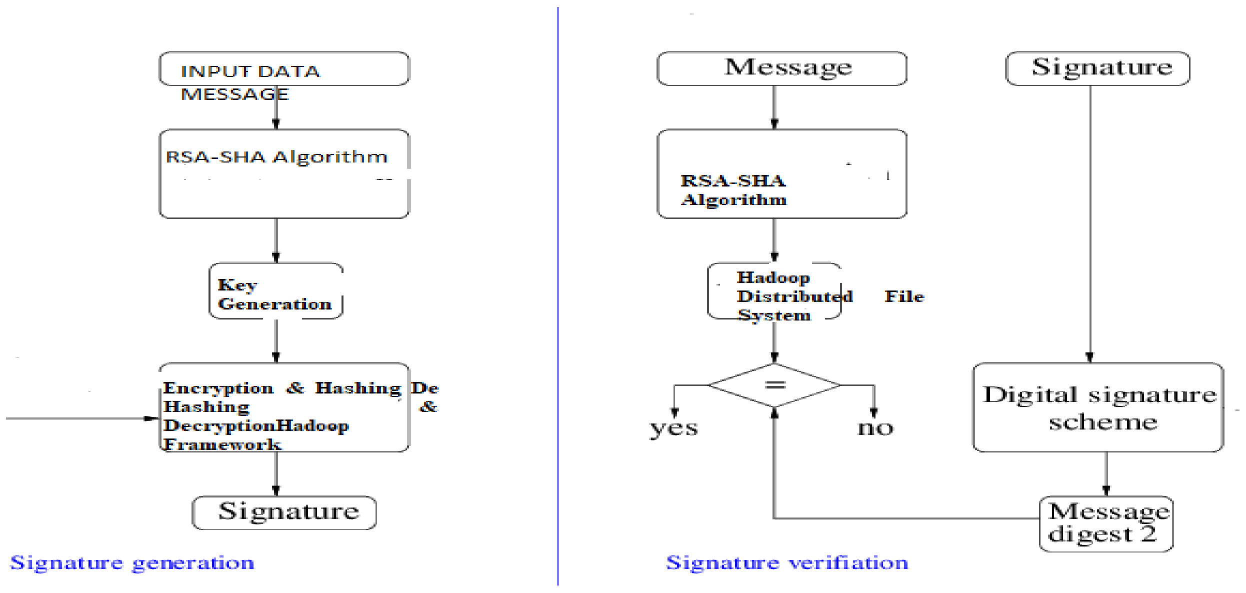


Fig. 3 .Flow diagrams of Signature Data

**D. RSA-SHA algorithm**

**Phase I: Prime Number Generation**

The creation of prime numbers serves as the starting point for the entire process in the proposed RSA-SHA algorithm based safe computing algorithm. Each new user U_i must register their login information. User ID (ID,PW) credentials are taken into account. The safe private and public keys are produced for every U_i. The process of creating prime numbers is originally started with this end in mind. The RSA-SHA algorithm starts the key creation process by creating prime numbers.

Each user must first choose the level of protection needed for the data. The security level is divided between high and low levels. In order to avoid more time consumption, the upper (_u) and lower (_L) boundaries are established. Depending on the security level of the user, the upper and lower boundaries are established. The _L will be low and the _u will be medium if the user just requires a low degree of protection. The _L is high and the _u is likewise high if the user demands a high level of security. The method is prevented from producing the lowest and relative highest prime integers by _u and _L.

Thus, the modified algorithm uses two boundary levels in order to improve the prime number generation process. This works upon the security level required by users. If the security level is high, then three large prime numbers are generated. Otherwise, two prime numbers are generated. In general, prime number generation is complex and time-consuming process.

**Phase II: Key Generation**

The prime numbers produced in the preceding step are used in the key generation phase. The key generation mechanism is adaptable and creates keys based on the user's desired level of security. The RSA-SHA technique creates keys with two prime integers when the user's security level is low. Otherwise, the RSA method generates keys using three prime integers. The procedure of creating keys based on security level progressively raises security level. Because "the attacker can easily crack the prime numbers from the public key and then derive the private key," this is the main problem with the RSA key generation technique. Therefore, the most crucial aspect of the RSA method is the key creation procedure.
The RSA-SHA method generates keys using a non-prime factor, in contrast to the traditional RSA algorithm. The inclusion of a non-prime element during the key generation process is the primary source of innovation for the RSA-SHA algorithm. All of the writers of RSA variations have focused on the prime factors; none of them have paid attention to non-prime factors. The following factors need to be taken into account when introducing non-prime numbers:A significant non-prime factor is required.It must be less than the prime factors for the non-prime factor to exist.
Along with the prime factors, a non-prime factor is chosen in the key generation procedure. Then, using the created prime and non-prime factors, it calculates the public key (Pu_k) and private key (Pv_k) for each user. The computation of $n$ value and $\varphi(n)$ values differs for users with different security level as follows,

$$n = \begin{cases} P * Q, & security\ level = low \\ P * Q * S, & security\ level = high \end{cases} \quad (1)$$

$$n = \begin{cases} (P-1) * (Q-1), & security\ level = low \\ (P-1) * (Q-1) * (S-1), & security\ level = high \end{cases} \quad (2)$$

Here $P, Q, S$ represents three large prime numbers that are generated by modified SoA algorithm as in pseudocode.3. The $Pu_k$ and $Pv_k$ are generated upon these $n$ and $\varphi(n)$. Similarly, the non-prime factor $(f)$ plays pivotal role in key generation phase. The $f \ll P, Q, R, S$. In this manner, the secret $Pu_k$ and $Pv_k$ are generated to improve the security level.

**Phase III: Encryption & Hashing**

In this phase, the user encrypts the random ID using the S(Pu_k) then, this encrypted random ID (En{R_ID^t (U_i )}) is sent to the corresponding user. Encrypting random ID assures that the random ID can be retrieved by only legitimate user who has the corresponding S(Pv_k). If an attacker gets this encrypted random ID, the attacker is unable to recover the original random ID without the private key.

Here, the HDFS server will get only a random ID and the public key of the user. As the HDFS is not fully trusted, the user's privacy credentials (ID and password) are not shared with it. Furthermore, the random ID varies over authentication requests. Thus, the user credentials are not fully shared with the server. Then the U_i generates the digital signature for the received random ID.

The Secure Hashing Algorithm is used by the U_i to compute the hash value for R_IDt (U_i) before producing Sign(R_IDt (U_i)). Padding, block deconstruction, and hash computation operations are used by the SHA to generate hashes. The data is initially padded to make sure it contains 512 bits. Data here stands for theSR_IDt (U_i). The data is then split up into message blocks (M), after which a hash value is created for each block. Finally, the following formula is used to calculate the data's hash value:
$H\left(R_{ID}^{t}(U_i)\right) = H^{j-1} + M\left(H^{(j-1)}\right)$(3)

The hash value is computed for each $j^{th}$ word in the message block $M$. The hash value generated is then signed by the private key of the $U_i$. For signing process, the $U_i$ encrypts the $H(R_{ID}^{t}(U_i))$ using $Pv_k$ as follows,

$\text{Sign}(R_{ID}^{t}(U_i) \rightarrow \left\{En\left(R_{ID}^{t}(U_i)\right) \| Pv_k\right\}$(4)

The user is allowed to upload or retrieve their data from the server. If the user has data $(d)$ to be stored in the server, then the user encrypts the data with the secure $Pu_k$. If the user needs to retrieve the data from the server, then the user retrieves the data and decrypts using $Pv_k$. The encryption and decryption process is performed as follows,
$En(d) = d^{e} mod\ n$(5)

Thus the data is secure before transmitting into the HDFS storage server and this ensures the privacy of the data.

**Phase IV: De Hashing & Decryption**

The digital signature verification process uses the $Pu_k$ to verify the legitimacy the user. The verification process is initiated with hash generation of $R_{ID}^{t}(U_i)$. The server first generates, $H\left(R_{ID}^{t}(U_i)\right)$ using SHA algorithm. Then, decrypts the received digital signature by using $Pu_k$ as follows,
$H\left(R_{ID}^{t}(U_i)\right) \leftarrow \{De\left(\text{Sign}\left(R_{ID}^{t}(U_i)\right)\right)\} \| Pu_k$(6)

The server further verifies the $H\left(R_{ID}^{t}(U_i)\right)$ generated by it and derived from the digital signature. If both hash values are matched, then the digital signature verification is successful. The user is authenticated to access the data. Otherwise, the user request is denied and the access is prevented. The user is provided with the security key. If the security strength is low in the previous phase, then the key regeneration process is initiated for the particular user. Here $(e, n)$ is the $Pu_k$ of the user $U_i$. Similarly, the decryption process is performed as follows,

$De(d) = \left(En(d)\right)^{d} mod\ n$(7)

Here $(d, n)$ is the $Pv_k$ of the user $U_i$. Thus, the data cannot be decrypted by any attackers. In this manner, the data is encrypted and decrypted in the cloud environment. Here, the data and keys are taken as small integers for simplicity. But the proposed RSA-SHA algorithm is suitable for large data also.

**E. Hadoop Framework**

. The processing power of Apache Hadoop is driven by the MapReduce programming language, while the storage power is provided by a Hadoop Distributed File System (HDFS).

Hadoop splits data into huge chunks and distributes it among cluster nodes. The bundled code is then sent to the node so that it may process the data concurrently. To generate the data set more quickly and efficiently than with a traditional computer chip architecture based on a simultaneous file system where the computation and data have been distributed via high-speed networking, this method uses data locality, or a portable node changing the data they have access to.

**A. Hadoop modules**

The fundamental parts of the Apache Hadoop framework are as follows:

Hadoop Distributed File System (HDFS) is a file system that is distributed across a cluster, allowing for extremely high aggregate bandwidth. Hadoop YARN is software that coordinates the allocation of computational resources in clusters and the scheduling of user applications. Hadoop MapReduce is software that uses the MapReduce framework to handle massive amounts of data.

**B. Hadoop Distributed File System**

Google's research and writing had a significant impact on the development of the HDFS and MapReduce modules for Apache Hadoop. Although there are also some C and shell script components, Hadoop is mostly written in Java. Thanks to Hadoop Streaming, an individual's application's "map" and "reduce" phases may be built in any language instead of just Java, which is frequently used for MapReduce. Several Hadoop ecosystem projects offer more comprehensive user interfaces.

Data in any particular Hadoop cluster may be quickly accessible thanks to HDFS, or the Hadoop Distributed File System.

It's a collection of items that aren't meant to be together. Since individual files may be many terabytes in size, HDFS is ideal for systems that must manage very huge datasets. In a master-slave architecture, NameNodes, which handle file system operations, assist DataNodes in managing storage arrays on individual compute nodes. HDFS was created to prevent data loss or corruption during software updates. HDFS I/O was created with these systems in mind because systems that perform batch processing, like Map Reduce, require excellent performance on successive reads and writes. Based on the relative distance between any two nodes, HDFS estimates the potential network bandwidth between them.

**C. Hadoop K-means clustering Map Reduce**

A particular type of computer cluster called a Hadoop cluster is made for the distributed preservation and processing of massive amounts of unstructured data. Low-cost commodity PCs are used in these clusters to provide power. since of its architecture, Hadoop clusters are frequently referred to as "shared nothing" systems since the only thing that the nodes in a cluster share is the network that connects them. Racks of servers are used to hold Hadoop clusters. Information transfer between nodes inside a single rack is preferred over information transfer between racks. The clustering approach has the inherent problem of forming clusters even when there aren't any clusters in the data by itself. Therefore, it is crucial to assess the clusters' inclination before any processing and then their density.
However, the number of clusters used for validation may differ (k). We do this by applying the validity indices for clustering introduced by Silhouette and Dunn to calculate the distance between groups and the distance within clusters. The recommended secure HDFS system includes the clustering method in Figure.4.

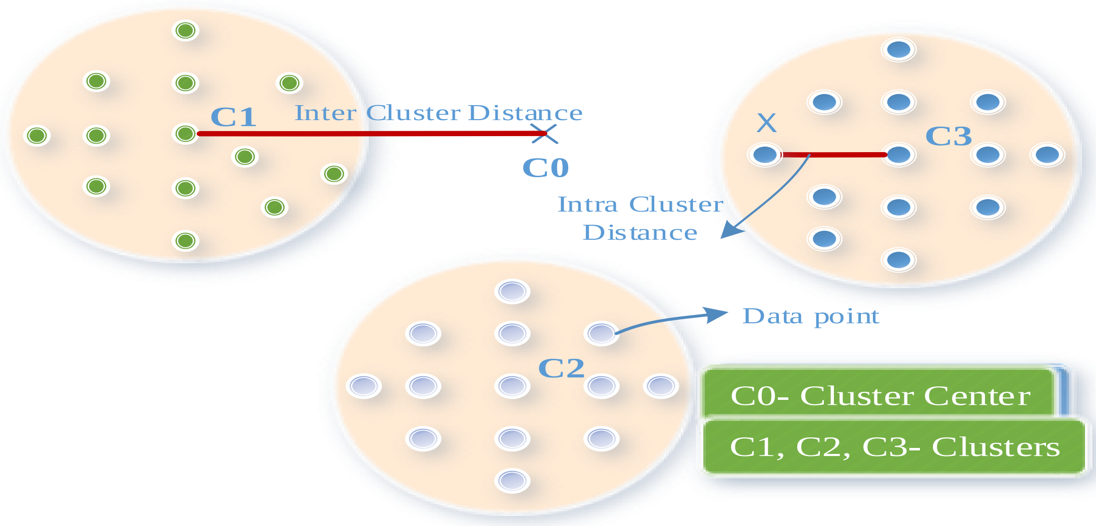


Fig.4. Hadoop clustering

In order to determine the distance between clusters, we first estimate the average distances between the centres of all clusters in the world $C_0$respectively, the centres of each cluster, which were represented as:

$$Inter\_Cluster = \frac{1}{n}\sum_{k=1}^{m} D(C_0, C_k)(8)$$

Intra-cluster distance is calculated between the ratio between the minimum of the distances from cluster centers and maximum of the distances from each point to its cluster center.

$$Intra - Cluster = \frac{\min_{k=1...n} D(C_k, C_0)}{\max_{k=1...N} \max_{X_i \in A_k} \{D(X_i, C_k)\}}(9)$$

In order to process enormous volumes of data, the Hadoop MapReduce framework was created. Map reduction frequently divides the input set of data into distinct sections, as seen in Fig. 5.

The mapping process is characterised by complete parallelism and a reduction in process-related tasks. Each cluster node has a master and slave process for the map. The indexing values are contained in the master, and the slave is in charge of the master's retrieval operation. While the master server manages all duties, each slave server maintains track of its own.
Below is a list of the fundamental map reduce entities.
• Master node - It oversees the jobs being monitored and enables users to post work requests.
• Slave node - It will always know where to be while the user is mapping, reordering, or reducing.
• Job tracker - By storing user schedules, it offers a method for allocating labour and keeping track of development.
• Task tracker - This device tracks the advancement of tasks and provides findings to the Job Tracker.
• Job - A thorough timetable has been supplied by the users.
. • Task - This is the breakdown of the task into its component pieces. For instance, a user task could be divided into several smaller tasks. All jobs employ the map & reduce function.
• Task effort - It is aware of the precise effort made in a slave node to complete an assignment.
We'll discuss the responsibilities involved in map reduction here.

## D. Map feature:

In the map phase, we first partitioned the data and used distance computation to conduct the Map process. Key and value are produced for partitioned data.The Figure 5 illustrated that Map reduced function bid data analyzed.
$Map\ (k_1, v_1) \rightarrow (k', v')^*(10)$

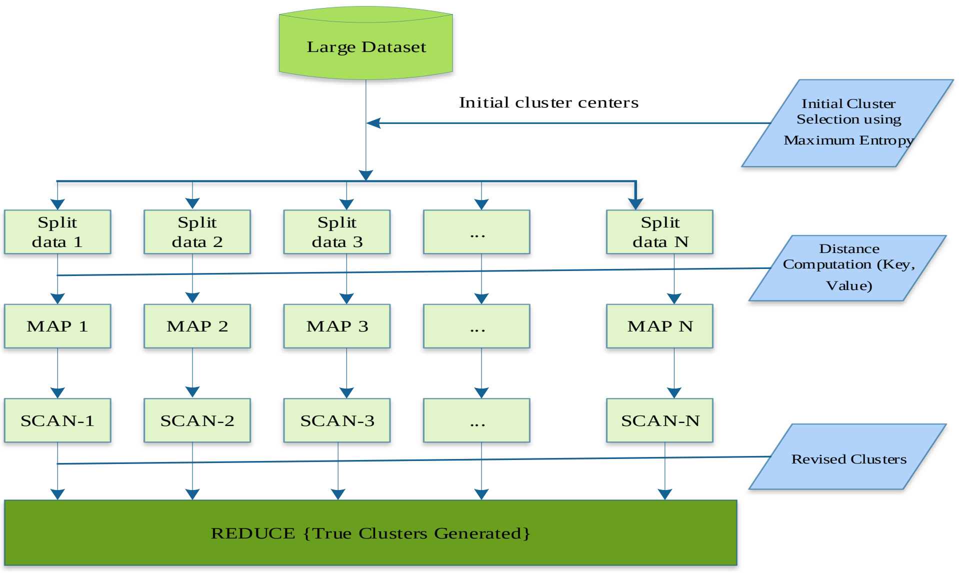


Fig.5. Map reduce function

## E. Reduce function:

Similar keys are clustered together, and the reduction function then finds the smallest possible set of high-quality clusters. Here are some of the ways in which it broadens the category of Reduce functions:

$Reduce\ (k', (v'))^* \rightarrow (k', v')^*(11)$

**Usually we can also indicate the whole process like**

$$Map\ (k_1, v_1) \rightarrow List\ (k_2, v_2)(12)$$

$Reduce\ (k_2, list\ (v_2)) \rightarrow List\ (k_3, v_3)(13)$

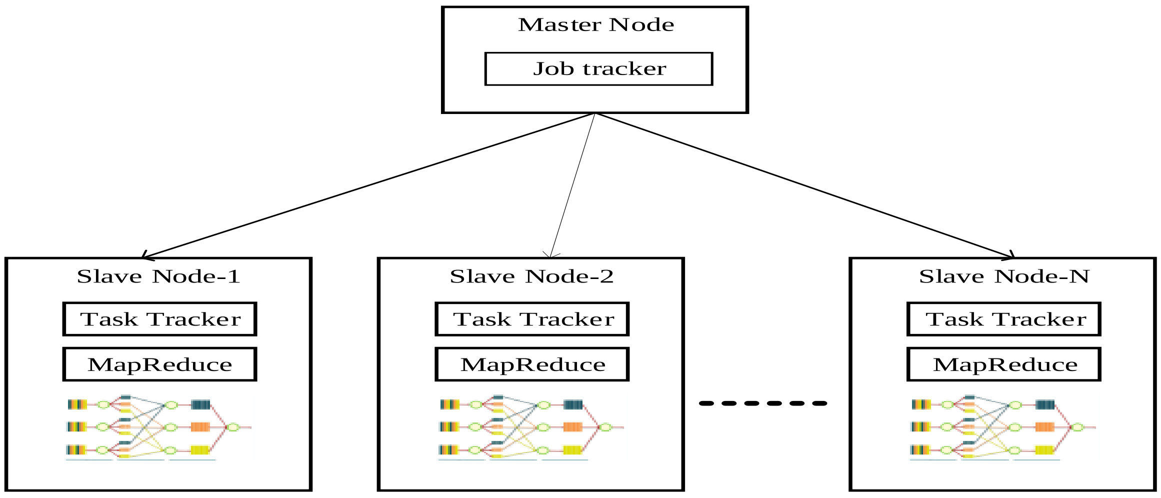


Fig.6. Master/slave communication model

The above figure 6 illustrated that Master slave communication model for big data analyzed.A single device controls a group of devices using the master/slave communication model. That is, under the master-slave paradigm, one node serves as the master, and the other nodes serve as the slaves. This notion was proposed to prevent the inconsistent data that was already present in the database. The slave node receives instructions from the master node and is given a task to complete. The work will be processed by the slave nodes, who will then send it back to the master. The master node is the controller in the majority of distributed systems that are active in real time, and its main responsibility is to oversee the duties assigned to each slave device that is connected to the controller. A new node is chosen as master by the slaves when the master node runs out of power or energy. Figure provides a comprehensive master/slave implementation design.6. Benefits of using a master/slave system

The ability to run backup services on the slave; the ability to perform real-time data analysis on information in the slave while simultaneously improving performance; the ability to replicate data so that it can be used repeatedly without involving the master; and the ability to distribute workload among numerous slaves to improve performance.

### F. Data Retrieval

The client request for our suggested system is first received as a URL and set of keys, which is sent to the master server in a name node, and the master node then sends the key to the slave node. If the key in the user request is valid and accessible, the slave server checks it and transmits it to the master server in name node. The data node that the master server delivers directly obtains the data from the server's header and uses the B+ tree structure to do so. The data is included in a B+ tree and is sorted according to the order of the key range. We retrieve the data and transmit it to the client using this B+ tree. The FIFO load balancing approach is used in this instance to distribute the workload across the servers according to the first-come-first-served principle. Figure.7 illustrates the steps required in data retrieval.

### G. Name node

We initially construct the name node, which houses the master and slave servers, in our proposed system's master-slave architecture. In this case, data retrieval is made efficient by using the Jetty server. The hash values of the related data are used to create the indexing keys for each name node server. The master server will create an index of all the data, and the slave servers will use the primary key discovered from the client's query to determine if the requested data is present in the index of the master server. If a match is found, the request is sent to the name node's master server, who subsequently sends it to the data blocks.

### H. Data node

The header and recorder are both components of the master and slave servers, which are part of the data node. When updating the recorder, the hear header serves as the B+ tree. The figure 7 illustrated that data retrieval method in big data Technique. The data node may simply access the data from the recorder using the key range values, which are [a-m, n-z, 0-9]. The recorder also includes the data. The slave server immediately retrieves the data after receiving the key values from the master node's name node and data node. Finally, the name node receives the key values and data.

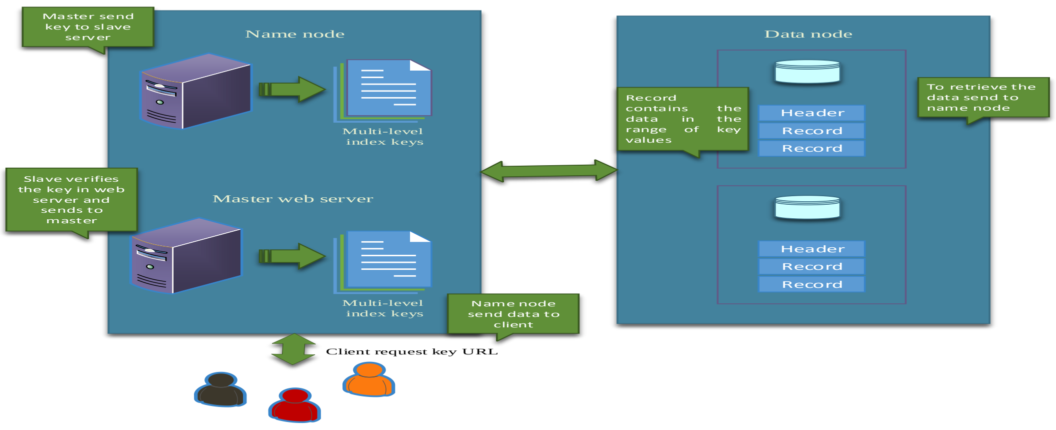

Master send key to slave server
Name node
Multi-level index keys
Slave verifies the key in web server and sends to master
Master web server
Multi-level index keys
Name node send data to client
Client request key URL
Data node
Record contains the data in the range of key values
Header
Record
Record
To retrieve the data send to name node
Header
Record
Record

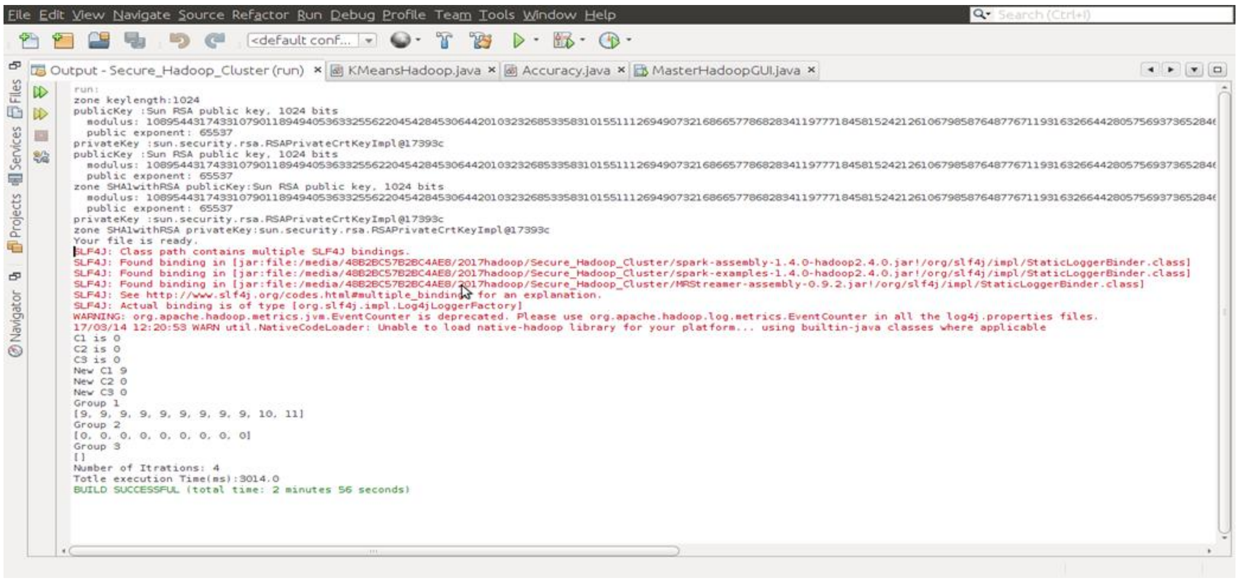
Fig.7.Data retrieval

## IV. Experimental results

Our proposed system runs with the Hadoop environment, Hadoop is the open source software frame work which stores the distributed data. Our work here proposes a new design with only two nodes—a name node and a data node—which significantly shortens the map-reduce paradigm's data-retrieval process. open source Simulation setup

- We simulate our proposed work with the Hadoop software environment; we configure the one name node and one data node.
- Here web applications deployed in HDFS [Hadoop Distributed File System].
- In order to distribute the workload between the two servers, we employ the apache 2 balancer. As a result of our efforts, both the masters and slave servers can handle the same amount of work.

For effective data retrieval across HDFS supervisors and slave nodes, use the Map Reduce operation, as seen in Figure 8. Each part was effectively clustered using the K-means algorithm.


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default conf...
Output - Secure_Hadoop_Cluster (run)
KMeansHadoop.java
Accuracy.java
MasterHadoopGUI.java
Files
Services
Projects
Navigator

```
run:
zone keylength:1024
publicKey :Sun RSA public key, 1024 bits
  modulus: 10895443174331079011894940536332556220454284530644201032326853358310155111269490732168665778682834119777184581524212610679858764877671193163266442805756937365284
  public exponent: 65537
privateKey :sun.security.rsa.RSAPrivateCrtKeyImpl@17393c
publicKey :Sun RSA public key, 1024 bits
  modulus: 10895443174331079011894940536332556220454284530644201032326853358310155111269490732168665778682834119777184581524212610679858764877671193163266442805756937365284
  public exponent: 65537
zone SHA1withRSA publicKey:Sun RSA public key, 1024 bits
  modulus: 10895443174331079011894940536332556220454284530644201032326853358310155111269490732168665778682834119777184581524212610679858764877671193163266442805756937365284
  public exponent: 65537
privateKey :sun.security.rsa.RSAPrivateCrtKeyImpl@17393c
zone SHA1withRSA privateKey:sun.security.rsa.RSAPrivateCrtKeyImpl@17393c
Your file is ready.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/media/48B2BC57B2BC4AE8/2017hadoop/Secure_Hadoop_Cluster/spark-assembly-1.4.0-hadoop2.4.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/media/48B2BC57B2BC4AE8/2017hadoop/Secure_Hadoop_Cluster/spark-examples-1.4.0-hadoop2.4.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/media/48B2BC57B2BC4AE8/2017hadoop/Secure_Hadoop_Cluster/MRStreamer-assembly-0.9.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated. Please use org.apache.hadoop.log.metrics.EventCounter in all the log4j.properties files.
17/03/14 12:20:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
C1 is 0
C2 is 0
C3 is 0
New C1 9
New C2 0
New C3 0
Group 1
[9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 11]
Group 2
[0, 0, 0, 0, 0, 0, 0, 0, 0]
Group 3
[]
Number of Itrations: 4
Totle execution Time(ms):3014.0
BUILD SUCCESSFUL (total time: 2 minutes 56 seconds)
```

Fig.8. Different Clusters

After four rounds, the clusters have been clustered into the three distinct clusters shown in Figure.9.

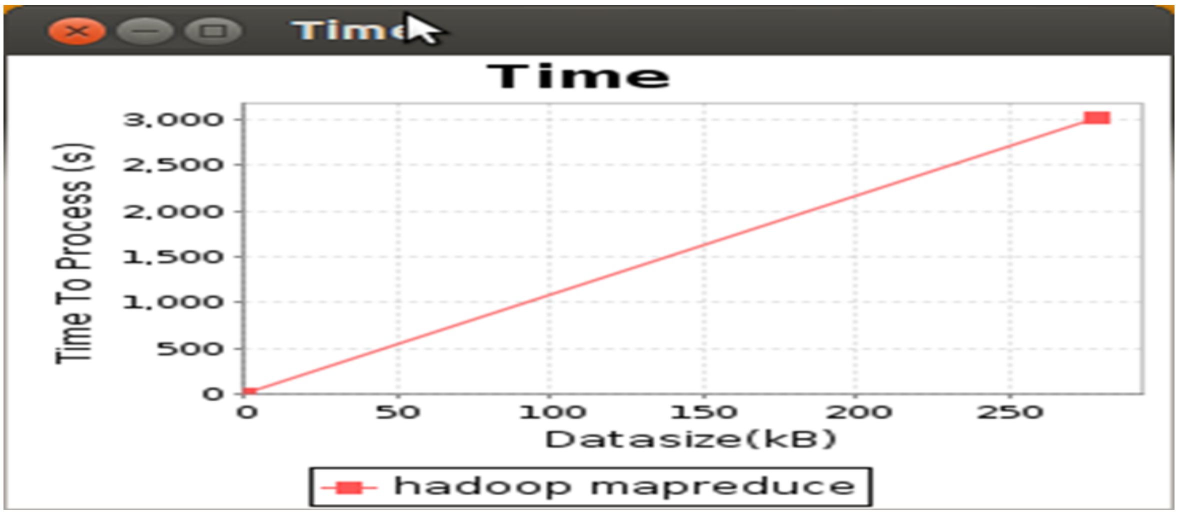


Fig.9. Graph Output

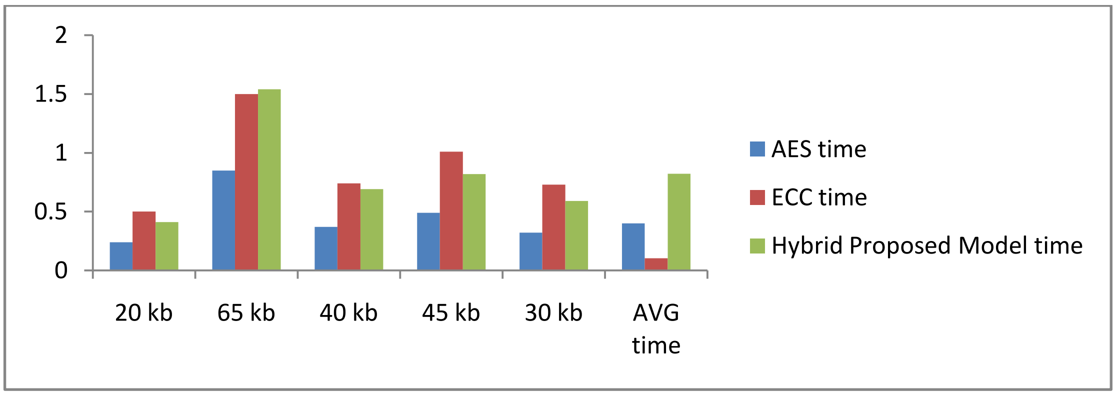


Figure 10. Final Results in AES Time and ECC time for Big Data

The final product of our proposed system appears in Figure.8, which depicts the time required to collect information from master and slave nodes. In this chart, we illustrate how the size of a file affects the time it takes to download. Now, we utilize the map-reduce paradigm to speed things up.The figure 9 illustrated that graph output for hadloop mapreduce method with the time process and datasize.

## V. CONCLUSION

The proposed work offers a safe Hadoop environment and a method for quickly getting data from Hadoop. Our HDFS-based chunks are encrypted using the RSA-SHA technique before being stored there. Between the master and slave servers, the RSA algorithm is doing the job of encrypting each piece, while SHA is checking the digital signature. Here, we propose a master-slave architecture with two nodes—a name node and a data node—and individual master-slave servers at each node. By using K Means Map Reduce, our suggested system improved performance and decreased data recovery time in HDFS.
In the near future, we want to implement a more effective clustering technique for Map Reduce. All in all, it makes our system run better.

## VI. **References**

I. Jam, M.R.; Khanli, L.M.; Javan, M.S.; Akbari, M.K., A survey on security of Hadoop, in the proceedings of the4th International eConference on Computer and Knowledge Engineering (ICCKE), 2014, vol., no., pp.716-721, 29-30 Oct. 2014.

II. Danish Shehzad, Zakir Khan, Hasan Dağ, Zeki Bozkuş,',"A Novel Hybrid Encryption Scheme to Ensure Hadoop Based Cloud Data Security"International Journal of Computer Science and Information Security (IJCSIS),Vol. 14, No. 4, April 2016.

III. T. Mohanraj,"Hybrid Encryption Algorithm for Big Data Security in the Hadoop Distributed File System,"Computer Assisted Methods in Engineering and Science, [S.l.], v. 29, n. 1–2, p. 33–48, jan. 2022. ISSN 2956-5839.

IV. Yalla et al. (2016) Yalla C, Gill A, Gupta M, Mohankumar H, McCloskey T, Minas L, Ngo N, Tolentino S, Watson D. Big Data: security Intel IT's apache hadoop platform. White paper, Intel. 2016.

V. Sudhakar, Farquad & Narshimha Sudhakar K, Farquad MAH, Narshimha G. Effective convolution method for privacy preserving in cloud over big data using map reduce framework. IET Software. 2019;13:187–194. doi: 10.1049/iet-sen.2018.5258.2019

VI. Meisuchi Naisuty,, Achmad Nizar Hidayanto, Nabila Clydea Harahap, Ahmad Rosyiq, Agus Suhanto, and George Michael Samuel Hartono,"Data protection on hadoop distributed file system by using encryption algorithms: a systematic literature review,The 8th Engineering International Conference 2019,1444 (2020) 012012 IOP Publishing,doi:10.1088/1742-6596/1444/1/01,2012.

VII. Vinay Poduval, Ashish Koul, Daniel Rebello, Karunesh Bhat, Revati M. Wahul. "Cloud based Secure Storage of Files using Hybrid Cryptography and Image Steganography", International Journal of RecentTechnology and Engineering (IJRTE)ISSN: 2277-3878, Volume-8 Issue-6, March 2020.

VIII. Bikash Agrawal, Raymond Hanse, Chunming Rong, Tomasz Wiktorski, "SD-HDFS: Secure Deletion in Hadoop Distributed File System", IEEE International Congress on Big Data, PP 181-189, 2016.

IX. Zhijiang Chen, Hanlin Zhang, William G. Hatcher, James Nguyen, Wei Yu, "A Streaming-Based Network Monitoring and Threat Detection System", 14th International Conference on Software Engineering Research, Management and Applications (SERA), pp 31-37, 2016.

X. Chaitali Gupta, Ranjan Sinha, Yong Zhang, "Eagle: User Profile-based Anomaly Detection for Securing Hadoop Clusters", IEEE International Conference on Big Data (Big Data), pp 1336-1343, 2016.

XI. Sharma, Vinod Kumar, Rajesh Singh, Anita Gehlot, Dharam Buddhi, Simone Braccio, Neeraj Priyadarshi, and Baseem Khan. "Imperative role of photovoltaic and concentrating solar power technologies towards renewable energy generation." International Journal of Photoenergy 2022 (2022).

XII. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in Operating system design and implementation (OSDI), a computer science conference, 2004, pages: 137–150.

XIII. Xingguo Cheng,Nanfeng Xiao,Faliang Huang "Research on HDFS-based Web Server Cluster", 2011 International conference on Digital object identifier, pages: 1-4.

XIV. Ramalingam, Rajakumar, Rajeswari Muniyan, Ankur Dumka, Devesh Pratap Singh, Heba G. Mohamed, Rajesh Singh, Divya Anand, and Irene Delgado Noya. "Routing Protocol for MANET Based on QoS-Aware Service Composition with Dynamic Secured Broker Selection." Electronics 11, no. 17 (2022): 2637.

XV. Zhang, H., Babar, M., Tariq, M.U., Jan, M.A., Menon, V., & Li, X. (2020). SafeCity: Toward Safe and Secured Data Management Design for IoT-Enabled Smart City Planning. IEEE Access, 8, 145256-145267.

XVI. Kumar, GS Arun, et al. "LoRa enabled Real-time Monitoring of Workers in Building Construction Site." International Journal of Electrical and Electronics Research 10.1 (2022): 41-50.

XVII. Hu, Y., Gunapati, V.Y., Zhao, P., Gordon, D., Wheeler, N.R., Hossain, M.A., Peshek, T.J., Bruckman, L., Zhang, G., & French, R.H. (2017). A Nonrelational Data Warehouse for the Analysis of Field and Laboratory Data From Multiple Heterogeneous Photovoltaic Test Sites. IEEE Journal of Photovoltaics, 7, 230-236.

XVIII. Babar, M., Khan, F., Iqbal, W., Yahya, A., Arif, F., Tan, Z.S., & Chuma, J. (2018). A Secured Data Management Scheme for Smart Societies in Industrial Internet of Things Environment. IEEE Access, 6, 43088-43099.

XIX. Aditham, S., & Ranganathan, N. (2018). A System Architecture for the Detection of Insider Attacks in Big Data Systems. IEEE Transactions on Dependable and Secure Computing, 15, 974-987.

XX. Bagwari, Swapnil, Anita Gehlot, Rajesh Singh, Neeraj Priyadarshi, and Baseem Khan. "Low-cost sensor-based and LoRaWAN opportunities for landslide monitoring systems on IoT platform: a review." IEEE Access 10 (2021): 7107-7127.

XXI. Hababeh, I., Gharaibeh, A., Nofal, S., & Khalil, I. (2019). An Integrated Methodology for Big Data Classification and Security for Improving Cloud Systems Data Mobility. IEEE Access, 7, 9153-9163.

XXII. Zhang, D., & Kabuka, M. (2018). Distributed Relationship Mining over Big Scholar Data. IEEE Transactions on Emerging Topics in Computing, 1-1.

XXIII. Dumka, Ankur, Parag Verma, Rajesh Singh, Anil Kumar Bisht, Divya Anand, Hani Moaiteq Aljahdali, Irene Delgado Noya, and Silvia Aparicio Obregón. "A Novel Deep Learning Based Healthcare Model for COVID-19 Pandemic Stress Analysis." Computers, Materials & Continua 72, no. 3 (2022): 6029-6044.

XXIV. Algaradi, T.S., & Rama, B. (2019). A Novel Blowfish Based-Algorithm To Improve Encryption Performance In Hadoop Using Mapreduce. International Journal of Scientific & Technology Research, 8, 2074-2081.