# Survey of Computation Integrity Methods For Big Data

Doaa Abo aly, Hamdy Mousa, Walid Atwa
Dept. of Computer Science, Faculty of Computers and Information
Menoufia University, Egypt

***Abstract*— Nowadays, big data becomes widespread. Big data has great value, but it faces many challenges. One of these challenges is security. Many classic security techniques exist, but these mechanisms are not appropriate for big data security. To secure big data, it is necessary to secure many aspects such as infrastructure, data privacy, data management, and integrity and reactive. Securing computations in distributed programming frameworks and protecting non-relational data stores are two requirements for infrastructure protection. This survey will highlight securing MapReduce as one of the most popular distributed programming frameworks. Security of MapReduce computation is an important consideration when a MapReduce computation is performed on a public or hybrid cloud. When a MapReduce job is executed on public cloud or hybrid cloud, an integrity check of its result is required. In this survey, a set of previous techniques that check the result integrity of MapReduce will be explained. In addition to discussion of the advantages and disadvantages of each technique.**

***Keywords—Big data, MapReduce, security, distributed computing.***

## I. INTRODUCTION

Big data is an expression that depicts the large structured and unstructured volume of data collected about our surroundings. Big data is a data set that is characterized by being big, high in variety, and velocity[1]. The advancement of information technology and social networks lead to the fast increase of data with the coming of the era of big data and cloud computing [2]. Many economic and political interests are existed in big data, especially the process of data integration, analysis, and data mining [3]. However, big data faces many security risks and privacy-preserving challenges[4]. The traditional security mechanisms are not able to deal with big data security. This is because of the velocity, volume, and variety of big data. One of these challenges is secure computations in distributed programming frameworks (DPFs). MapReduce is a popular example for DPFs[5]. MapReduce presents parallel processing of large-scale data[6]. It is used to simplify distributed processing of large scale data in an efficient and fault-tolerant manner on a public cloud or hybrid cloud without any overprice (MapReduce is presented as platform-as-service by cloud) [7]. Without taking into consideration physical infrastructures and installation of software, there are more public clouds (*e.g.*, Amazon Elastic MapReduce, Google App Engine) that enable users to complete MapReduce computations. MapReduce on both public cloud and hybrid cloud suffers from many attacks and security threats. The users can cost-effectively process big data using MapReduce on public clouds. But MapReduce on public cloud or hybrid cloud faces integrity vulnerability problem. If the public cloud is evaded because of security problems and running everything on private cloud, this will achieve result accuracy but with less economic benefit. On the public cloud, MapReduce jobs are executed by a cluster of hundreds or thousands of computation nodes. If an impersonation attack dominated any worker on this cluster, it will control this worker and make it tampers with computations. This worker becomes a malicious worker and may generate the wrong results and the total result of computation becomes incorrect. So it is necessary to check the result integrity of all workers whether mapper or reducer to satisfy the integrity of computations requirement and eliminate malicious adversary. In addition, more applications (e.g., Google, Yahoo!, Facebook, etc.) use MapReduce to process data, so the result integrity of MapReduce computation must be ensured. In this survey, a set of techniques that try to check the result integrity of MapReduce computation will be explained and their point of weakens will be clarified.

The remaining of the survey is showed as follows. The model of MapReduce data processing is introduced in Section II. In Section III, a set of previous techniques that check the result integrity of MapReduce will be explained. In section IV, the problems of these techniques and the future work are showed. Finally, Section V concludes the survey.

## II. MAPREDUCE FRAMEWORK

MapReduce presents parallel processing using computing nodes cluster over large-scale data as shown in Fig.1. Three entities are components of MapReduce [8]: distributed file system (DFS), a master, and workers. DFS where data are stored in the distributed file system in the form of a data block. The master presents job management, task scheduling, and load balancing among

workers. Workers are computation resources that perform tasks specified by the master. MapReduce contains two phases: 1) a map phase, in this phase; parallel processing is done by distributing input data to different distributed workers. 2) Reduce phase will be collected the intermediate results together. The master takes the job of MapReduce from users. This job's input text files will be put in DFS in the data blocks form. This job is partitioned into several map and reduce tasks. Map tasks number is determined by how many data blocks the input text files include. Only one data block will be taken by each map task as its input. In the map phase: when the master assigns map task to a worker, a worker becomes a mapper. When the map task assignment is sent by the master to the mapper, the data block is read by the mapper from DFS. Then mapper processes the data block and stores its intermediate result in its local storage. Each mapper generates an intermediate result that is partitioned into $k$ partitions $p_1, p_2, \ldots, p_k$ by partitioning function. Reduce tasks number is equal to partitions number k. In reduce phase, when the master assigns reduce task to a worker, a worker becomes a reducer. After a reduce task is received by the reducer, the master will send a notification when a map task is completed. After this notification, the reducer will read the intermediate result of each mapper that ends its map task. Then, the reducer processes its partition that is read from the intermediate result. Finally, each reducer result will be written to the DFS.

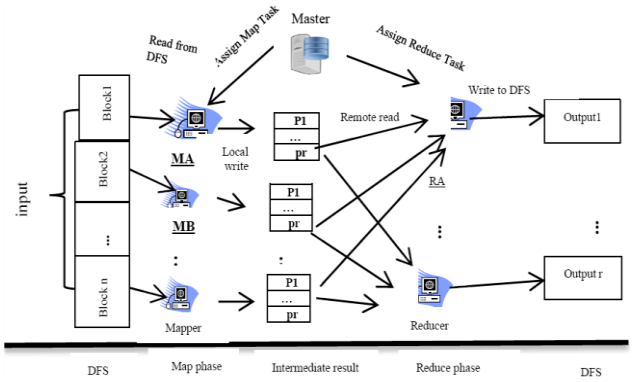


Fig. 1. The mapreduce data processing reference model [8].

## III. COMPUTATION INTEGRITY TECHNIQUES IN BIG DATA

MapReduce's importance stems from its simple paradigm and parallel processing capability for data-intensive computation in a variety of applications and research fields. Some researches interested in how to use MapReduce to fix problems in specific application domains. Other researches (that will be presented in this survey) are interested in computation integrity protection for MapReduce. To address computation integrity problems for MapReduce, some solutions are proposed such as replication sampling, and verification techniques. Some of these techniques achieve high result integrity but incur high-performance overhead and other techniques achieve low-performance overhead, but they are more vulnerable to attack as shown in Table.1.

**Table.1.overhead and result integrity of each technique**

| Technique name | Result integrity | | Overhead | |
|---|---|---|---|---|
| | low | high | low | high |
| secure MR | √ | | √ | |
| CCMR | | √ | | √ |
| Integrity MR | | √ | | √ |
| Accountable MapReduce | | √ | | √ |
| TrustMR | √ | | √ | |
| Result Verification Mechanism | √ | | | √ |
| MtMR | √ | | | √ |
| Credibility-Based Result Verification for MapReduce | √ | | | √ |

### A. *Secure MapReduce (secure MR)*

Secure MR provides MapReduce service integrity and prevents replay and denial of service attacks [8]. It adds a set of security components for MapReduce as shown in Fig.2 to check the result integrity of map/reduce tasks.

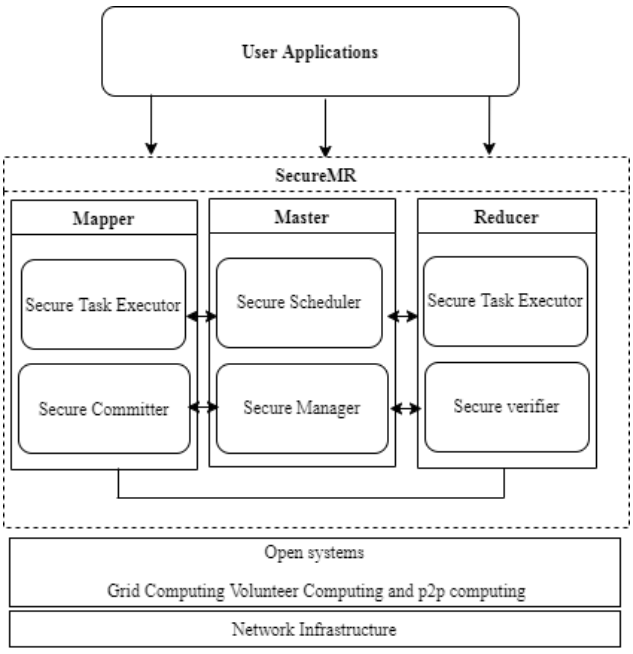


Figure.2 SecureMR Architecture Design [8]

Communication between entities in secure MR with each other to achieve MapReduce security protection is organized into two protocols:-

1. *Commitment protocol*

In this protocol, instead of sending intermediate results directly (that is expensive) to the master to

check the integrity of this result, mappers will only send commitments to the master to detect opposition as shown in Fig.3.

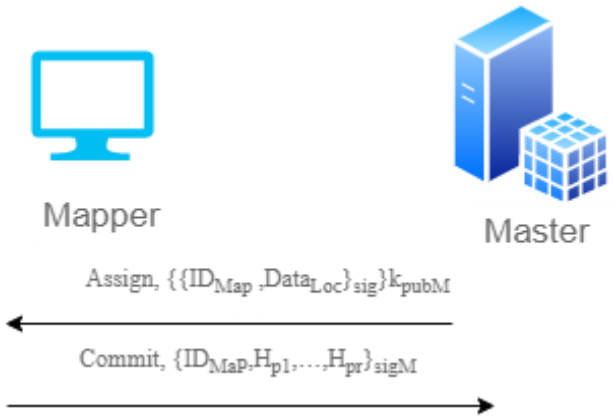


Fig. 3. The Commitment Protocol [8].

*2. Verification protocol*

In the protocol of commitment, mappers probably send to the master right commitment but send the wrong result to reducers. This problem is solved by verification protocol through asking reducers to ensure that the result and its commitment are in agreement as shown in Fig.4.

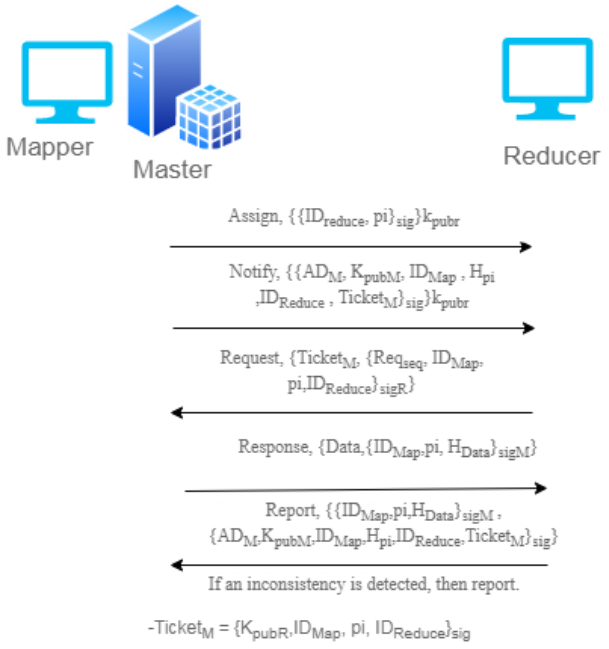


Fig. 4. The Verification Protocol [8].

This technique achieves service integrity of data processing with imposing low-performance overhead but it is not able to detect collusive malicious workers [8].

*B. Cross Cloud MapReduce (CCMR)*

CCMR merges the benefits of private cloud and public cloud [9]. It applies on MapReduce framework on a hybrid cloud that contains one private cloud (consist of master and verifiers) and one public cloud (consists of DFS and slave workers) as shown in Fig.5. It provides a scheme of checking result integrity on both the map phase and reduce phase. This technique has three types of tasks in both map and reduce phases (original task, replication task, verification task). In CCMR, a task will be executed on a worker that is randomly chosen from the public cloud (original task). Then, this task will be replicated on another worker that is also randomly taken from the public cloud (replication task). Master will compare between the original task and replication task to validate the result of the original task. Finally, this task will be executed on a private cloud to verify the result returned by the replication task and prevent collusive between two malicious workers that may execute both the original task and replication task on the public cloud. This technique ensures high result integrity but it causes non-negligible performance overhead.

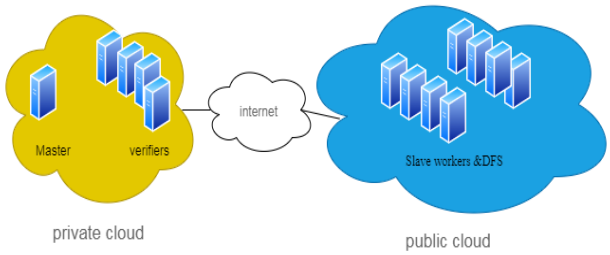


Figure .5.The architecture of CCMR [9].

*C. Integrity MapReduce (Integrity MR)*

To satisfy the needs of integrity MR, will use CCMR (previous technique). Where integrity MR is the same architecture of CCMR but it contains one private cloud and multiple public clouds as shown in Fig. 6. The original task will be executed by a randomly chosen worker from one of public clouds and the replication task will be executed by another worker that will be randomly selected from another public cloud which is not the same public cloud that is used in the original task [10]. Finally, the verification task will be executed in the private cloud. The comparison between three tasks will be done such as CCMR. This technique achieves high integrity with a non-negligible performance overhead.

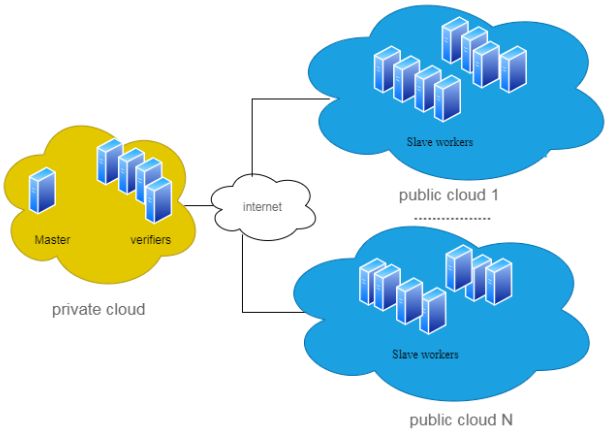


Figure.6. Architecture of Integrity MR [10].

*D. Accountable MapReduce*

In accountable MapReduce, each machine is responsible for its behavior [11]. It will detect malicious workers through the master submits the input data and output of each mapper and reducer to the auditor group (trustworthy nodes) to check the validity of each worker as shown in Fig.7. In the auditor group, the group member will repeat the processing of the input data and compare its output with the original one for inconsistency check. To reduce overhead, accountable MapReduce allows the system to apply p-accountability that reduces the records that need to be checked.

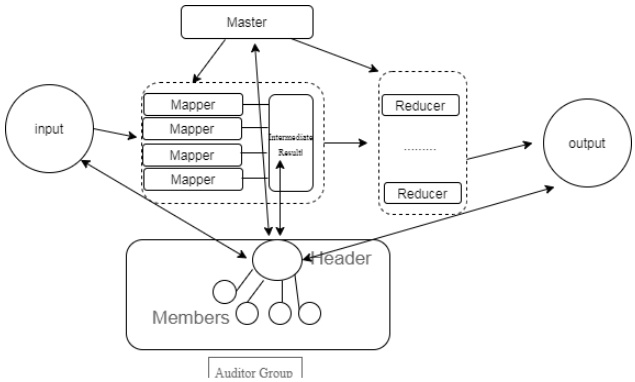


Figure.7. Accountability test [11].

*E. Trust MapReduce (TrustMR)*

Both map and reduce tasks are divided into smaller computation pieces. To ensure the result integrity of each mapper, TrustMR depends on replicating the subset of pieces of computation two times [12]. Then the full original task and its replicated subsets will be sent to map verifiers that are located on reduce to validate the result of the mapper before submitting it to the reducer. In reduce phase, various reduce functions are produced for every different key by reduce tasks. Some of these reduce functions are (randomly) chosen and replicated in the replicated reduce tasks by TrustMR. The original and replicated reduce functions are verified in the verifier of the reducer. This technique succeeds in achieving a high detection rate with decreasing the replication overhead but the map verifier that checks result integrity of the intermediate result may be malicious and provide wrong input to the reducer.

*F. Result Verification Mechanism*

A new technique is proposed based on reputation-threshold value using the method that is called voting [13]. In this technique, every task will be repeated and executed on various workers, and their outputs are gathered into sets based on outputs value. A worker's reputation is the total computing behavior of each worker. In the map phase, available mappers receive $N$ mapper tasks ($M_{T1}$, $M_{T2}$... $M_{TN}$) from the master to execute these tasks. After execution, mappers will send their outputs to reducers. For simplicity, only one task ($M_{T1}$) will be repeated $r_p$ times. Various mappers $P_i$ ($1 \leq i \leq r_p$) will receive these replicas to execute them. The $r_p$ intermediate results that may have various values $V(P_i)$ will be collected by the reducer. Then these results will be classified into $k$ result groups $G_j$ ($1 \leq j \leq k$) by the reducer. $R(P_i)$ refers to the reputation of mappers $P_i$. Every result group reputation $R(G_j)$ will be calculated as the reputations sum of all mappers that produce the same result value, noted $V(G_j)$. The reputations sum of all $r_p$ mappers:

$$R(Gj) = \frac{\sum_{i=1}^{rp}\{R(Pi): V(Pi) = V(Gj)\}}{\sum_{i=1}^{rp} R(Pi)}$$

When only one mapper task replica is sent, the reducer checks that the obtained result is true if the mapper reputation surpasses the minimum score of worker ($R(P_1) > WS$), and agrees on it as a reduce task inputs. Otherwise, more replicas of the task will be requested and reputation-threshold-based voting will be applied. When a new result for a task replica is received by the reducer, this result will be added to the group of results that has the same result value. Every the reputation of result group is computed, and the group of results that has maximum reputation $\max_j(R(G_j))$ will be picked as the best result group. If $\max_j(R(G_j))>\lambda$, the result group which performs this maximum is considered as the group, so this group result of mappers is agreed as right, and this result is taken by the reducer as input for its task. Finally, the reputation value of these mappers is updated as follows:

$$R(Pi) = \frac{n(Pi) + 1}{N(Pi) + 2}$$

Where $N(Pi)$ refers to the total number of tasks processed and $n(Pi)$ refers to the number of correct results produced by that mapper.

In reduce phase, the same above method is applied, but the master is instead of the reducer in the map phase.

*G. MapReduce Computation Integrity with Merkle Tree-based Verifications (MtMR)*

MtMR applies on hybrid cloud that contains one private cloud (contains of master and verifiers) and one public cloud (contains of DFS and slave workers). MtMR contains the checker that is the private cloud and the prover that is the public cloud [14]. The checker wants to check if the result executed by the prover is right or not. The verification is executed in a commit-and-verify manner. It is divided into four steps. The job that is executed on the prover is divided to *n* parts. The result of the $i_{th}$ part is $f(x_i)$ Where $f(x_i)$ is the computation result of the input data $x_i$.

1. *The commit step:*

The prover forms a Merkle tree based on the result $f(x_i)$, where $i \in [1,n]$. After the $\Phi$ value of the root node $R$ is calculated, it is sent to the checker by the prover as commitment. Where the Merkle tree's root node value $\Phi$ is returned by $\wedge(x_1, x_2\ldots, x_n)$ if the leaf nodes of Merkle tree are $\{x_1, x_2, \ldots, x_n\}$.

$$\Phi(Li) = f(xi), i \in [1, n]$$
$$\wedge(x_1,\ldots,x_n)=\Phi(R)$$

2. *The challenge step:*

The checker then takes some values from all the $f(x)$ values to verify (re-computing) those values. Taken values are referred as $f(x_j)$, where $j \in [1, m]$. The checker requests from the prover the $\Phi$ values that belong to the complementary nodes for every taken value $f(x_j)$, so that the Merkle tree's root value Φ depended on each sample $f(x_j)$ and its corresponding complementary nodes is regenerated by the checker.

3. *The prove step:*

After the prover had received a request from the checker, it sends the complementary nodes of every sampled $f(x_j)$ value. Complementary nodes of every taken value $f(x_j)$ is called as the proving path, referred as $\lambda_j$.

4. *The verify step:*

For each taken value $f(x_j)$ and its proving path $\lambda_j$ , the checker will recompute the $\Phi$ value of the root node, referred as $\Phi(R')$. Re-computation is carried out in a function $\wedge(f(x_j), \lambda_j)$, which is a chain of hash functions on $f(x_j)$ and elements in $\{\lambda_{j1}, \lambda_{j2}\ldots \lambda_{jk}\}$:

$$\wedge(f(x_j),\lambda_j) = hash(\ldots\ldots hash(f(x_j), \lambda_{j1}), \ldots,\lambda_{jk})$$

If the resulting Φ(R') is not the same Φ(R), the checker ensures that the taken nodes values from the prover are incorrect.

*H. Credibility-Based Result Verification for MapReduce*

The credibility of nodes and results is the factor for result verification in this scheme [15]. The result that has a high credibility value is taken over the result that has relatively low credibility. Quiz jobs and Map-reduce jobs are types of jobs that from through deciding the credibility of a node.

*a. Quiz Jobs*

In this scheme, each quiz job consists of a random number of map-reduce jobs with recognized results. The client delivers these jobs at random intervals of time. After each job of the quiz is executed, the hash value of the results are calculated, then the client receives these hash results. The client updates the credibility of the nodes.

*b. Map-reduce Jobs*

After the task is assigned to workers, these workers will return the hash of the calculated results. Frequency, credibility, and threshold count are computed for each distinct result *h*. Result that has maximum frequency and maximum average credibility is the correct result. If multiple results with the maximum frequency and maximum credibility exist, then an additional parameter that is called threshold count will be chosen to determine the right results.

## IV. Discussion

This survey presents some techniques that check result integrity of MapReduce computation to ensure result accuracy. Some of these techniques achieve high result integrity but incur high-performance overhead. In addition, others achieve low-performance overhead, but they are more vulnerable to attack. So in future work, the researchers must interest to reduce the performance overhead and achieve high result integrity for the techniques that check result integrity of MapReduce.

## V. CONCLUSION

Big data has useful information, but it also brings many challenges. One of these challenges is security of big data. To secure big data, there are more of aspects that should be secured such as infrastructure security. To secure infrastructure, it is necessary to secure computations in distributed programming frameworks and non-relational data Stores. This survey highlights on securing a popular example of computations in distributed programming frameworks, which is MapReduce. To secure MapReduce, each node that participates in MapReduce computation will be checked to ensure result integrity of this node. This survey presents some techniques that check the result integrity of MapReduce computation to ensure the result accuracy.

## References


[1] J. Wang, Y. Yang, T. Wang and J. Zhang, "Big Data Service Architecture: A Survey," Journal of Internet Technology, vol. 21, no. 1, pp. 393-405, 2020.

[2] G. M. Vaidya and M. M. Kshirsagar, "A Survey of Algorithms, Technologies and Issues in Big Data Analytics and Applications," in 4th International Conference on Intelligent, 2020.

[3] W. Fang, X. Z. Wen, Y. Zheng and M. Zhou, "A survey of big data security and privacy preserving," IETE Technical Review, vol. 34, no. 3, pp. 544-560, 2016.

[4]" Cloud Security Alliance. Expanded top ten big data security and privacy challenges," [Online]https://downloads.cloudsecurityalliance.org/initiatives/bdwg/, 2013.

[5] D. N, S. B and V. S, "Big data Hadoop MapReduce job scheduling: A short survey.," Information Systems Design and Intelligent Applications, no. 5, pp. 349-365, 2019.

[6] S. N, "Integrated Security and Privacy Framework for Big Data in Hadoop MapReduce Framework," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, no. 6, pp. 646-662, 2021.

[7] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in the 6th Conference on Symposium on Operating Systems Design & Implementation, 2004.

[8] W. Wei, J. Du, T. Yu and X. Gu, "Securemr: A service integrity assurance framework for mapreduce," in Annual Computer Security Applications Conference, 2009.

[9] Y. Wang, J. Wei and M. Srivatsa, "Result integrity check for mapreduce computation on hybrid clouds," in Sixth International Conference on Cloud Computing, 2013.

[10]Y. Wang, J. Wei, M. Srivatsa, Y. Duan and W. Du, "IntegrityMR: Integrity assurance framework for big data analytics and management applications," in International Conference on Big Data, 2013.

[11] Z. Xiao and Y. Xiao, "Achieving accountable MapReduce in cloud computing," Future Generation Computer Systems, vol. 30, pp. 1-13, 2014.

[12] H. Ulusoy, M. Kantarcioglu and E. Pattuk, "TrustMR: Computation integrity assurance system for MapReduce," in International Conference on Big Data (Big Data), 2015.

[13] A. Bendahmane, H. Bennasar and M. Essaaidi, "An efficient approach to improve security for mapreduce computation in cloud system," in the International Conference on Learning and Optimization Algorithms: Theory and Applications, 2018.

[14] Y. Wang, Y. Shen and H. Wang, "Mtmr: Ensuring mapreduce computation integrity with merkle tree-based verifications," IEEE Transactions on Big Data, vol. 4, pp. 418-431, 2016.

[15] T. A. Samuel and A. N. M, "Credibility-based result verification for Map-reduce," in Annual IEEE India Conference (INDICON), 2014..