Provable Data Possession Scheme Based On Algebraic Signature And Linked List For Outsourced Dynamic Data On Cloud Storage

Yasir Ali Panhwer, Imtiaz Ali Brohi, NajmaImtiaz Ali,Fiaz Ahmed Memon, Shahzad Nasim

Abstract: With the advancements of emerging technology and applications, the production of data is increasing rapidly and it has the property of being updated dynamically. These dynamic data is stored on cloud storage provided by third party service providers (CSP). The third party cloud storages cannot be solely trusted, hence the integrity of data is a major concern for data owners as if their data is being intact, not deleted, modified or destroyed intentionally and unintentionally. Several researchers have proposed various provable data possession (PDP) techniques provide probabilistic approach for data integrity in which integrity of data is verified at block level. However, due to enormous amount of data and its dynamic nature, the integrity verification schemes cause high computational cost and communicational overhead for the generation of metadata and node rebalancing of the data structures. In this paper, we propose a Modular PDPthat is based on Algebraic signature and linked list. The algebraic signature is used to calculate the signature for data blocks which have the computational complexity of$O(N)$. Moreover, the overall integrity verification process is also computationally efficient of$O(N)$. Furthermore, the communicational overhead for verification require only 256 bits hash of aggregated data blocks and 256 bits of corresponding aggregated signature blocks of data exchange between data owner and CSP. The linked list dynamic data operation such as delete, update, insert and append have efficient computational expenses of $O(n), O(1), O(1), O(1)$ respectively. The results shows that there is an increase of performance by 60% for data delete and insert operations as compared to iTable.

Index Terms: Cloud Computing, Data Integrity, Dynamic Data, Linked List, Provable Data Possession

1 INTRODUCTION

In the 21st century age of information technology, data is growing exponentially day-by-day. That exponentially growing data is termed as big data which specifically refer to data sets that are large, complex, structured and non-structured produced by day-to-day business [1]. This increasingly huge amount of dynamic data is part of day-to-day business that is being produced by technological advanced application such as internet, social networking sites, healthcare application, sensor networks and various companies. Thus, storing, processing and maintaining security of data on the local storages is a great challenge for companies and individuals because it incur high expenses for the infrastructure to store and process the big data [2]. In consequences, this dynamically changing big data requires excessive powers for processing and storage. Therefore, this power is provided by an emerging technology known as “Cloud Computing”.

Cloud computing is the trending technology in the world of computing, which is emerging with broad ranging effects across IT, business, health care, software engineering and data storage [3]. Cloud computing, mainly provides three service delivery model to its customers, which are “Infrastructure as a service (IaaS), Software as a Service (SaaS), and Platform as a service (PaaS) with four deployment models that is private, public, hybrid, and community cloud” defined by the National Institute of standard and technology (NIST) [3]. Cloud computing implements the virtualization technique to efficiently provide resources to end users. The main characteristics of cloud computing is that it provides on demand service, resource pooling, highly scalable, flexible, low cost computational powers for application, storage and platforms[4]. The major cloud services are used for storing data, sharing data and application services. Most of the enterprise companies are motivated towards the cloud computing, hence moving their application development, data storage (financial, personnel, healthcare) towards cloud services. Confidentiality, integrity, availability are some core security services that are required from the cloud storage for storing and processing organizational information. However, the primary focus for this research project is verification of data integrity as the correct security measures to be adopted by the Cloud Storage is a serious challenge [5]. Data integrity is defined as the sense of assurance that data is accurate and consistent throughout its entire life cycle. Further data integrity insures that the data is intact that is, not lost or damaged intentionally or accidentally[4].Integrity of data is intensively been researched for many years and integrity verification can be classified into two major approaches:

i) Deterministic approach in which whole file is processed for the integrity verification[6]. It guarantees 100% possession of data and is good for checking the small sized data.

ii) Probabilistic approach where only required blocks are checked to ensure the integrity of file[6].It might not provide 100% assurance of the data correctness, but it is viable for big data files on cloud storage.

Traditionally, the simplest way to assure the integrity of the data is to compute Message authentication code (MAC) before outsourcing the file to remote cloud storage. The data owner saves the MAC of the file on the local drive for verifying the integrity of the file. When the integrity of data is challenged, the verifier sends a request to retrieve the file from cloud storage. Then the verifier re-computes the MAC of the outsourced file. It compare the local stored MAC with the outsourced re-computed MAC to verify the integrity of data

- *Yasir Ali is currently Lecturer at Department of Information & Communication Technologies, Begum Nusrat Bhutto Women University, Sukkur, Pakistan. E-mail: yasir.panhwer@bnbwu.edu.pk.*
- *Imtiaz Ali Brohi is currently Associate Professor at Department of Information & Communication Technologies, Begum Nusrat Bhutto Women University, Sukkur, Pakistan. E-mail: imtiaz.brohi@bnbwu.edu.pk.*
- *NajmaImtiaz Ali, Fiaz Ahmed Memon are Assistant Professor at Institute of Mathematics & Computer Science, University of Sindh, Jamshoro.*
- *Shahzad Nasim is Associate Professor at Department of Management Sciences & Technology, Begum Nusrat Bhutto Women University, Sukkur, Pakistan. E-mail: shahzadnasim@live.com*

[7]). However, more feasible approach is that first data owner divides the file into n blocks, then computes the MAC of each block with a secret key. The owner sends the file and the MAC to cloud server and deletes the computed MAC and file itself from local storage. The data owner only stores the secret key. For verification the verifier requests the file block and the corresponding MAC from the remote server. The verifier computes the MAC with the secret key and compares it with the corresponding received MAC from the server. The above approaches can work great when the data size is small and the data is not changed frequently after being stored on the server. However for the cloud data both approaches have serious flaws and are impractical. In the first approach it have high communication cost such as if the 10GB or 100GB file is outsourced so every time to verify the integrity the data owner need to download the file from the cloud storage which is impractical because of bandwidth and internet data usage. While with the second approach cannot cover the nature of dynamic updates of data. In conclusion, due to high amount of data the above approaches have severe disadvantage as they have high communicational and computational cost. Moreover, it does not support the integrity checking of dynamic nature of data [8].

2 RELATED WORK

Remote data possession schemes are of two types, which are provable data possession (PDP) and proof of retrievability (POR) to verify the integrity of data on the cloud storage. PDP and POR use the technique of sampling random blocks of a file rather than reading the whole file for the verification. In both schemes the metadata (metadata is a pre-processed data based on the original data) is stored on cloud storage with the original data and it is used to verify the integrity of the data on demand [9]. The major difference between PDP and POR is that PDP only verify the correctness of data that the cloud service provider is holding the data but the POR provides the data recovery as well. POR incorporate erasure or error correcting codes to recover the damaged data at cloud storage [10]. This research primarily focuses on the PDP schemes because PDP is the first step for data integrity verification followed by POR to correct the damaged data. Several provable data possession schemes are proposed by different researchers but they lack to provide the efficient data integrity verification due to exponential calculation for generation of metadata [11], [12]. Furthermore, due to the rebalancing of the data structures; the current PDP's does not cater efficiently the dynamic nature of data on the cloud storages The first Probabilistic secure PDP was introduced by [13]) which can verify the integrity of static data without downloading the data from the cloud storage. The protocol uses RSA based homomorphic verifiable tags (HVT) based metadata generation for integrity verification. The protocol follows the client server structure and verification is done when the client send the challenge request to server and in response the server sends back the proof that is verified by the data owner or verifier. The drawbacks of this protocol are that, it incurs high computational powers for metadata generation because of using RSA exponential calculations and does not support the data dynamics property of outsourced dynamic data on cloud storage. The work of [13] has been extended by [14] to support the dynamic data property for outsourced data on cloud storages. It uses the HVT for generation of metadata for integrity verification. The dynamic data updates are supported by using authenticated skip list data structures. The weakness for this protocol follows high computational expenses for metadata generation. The dynamic updates also have the limitations as for every dynamic update the nodes of the skip list needs to be updated which incurs high computations. An improved homomorphic verifiable tag based scheme was proposed by [7] to support the data dynamics. They used the merkle hash tree to support the data dynamics. The root of the merkle hash tree is the aggregate hash of the lead nodes and the leaf nodes contains the hash of the tags. The weakness of the protocol is that it creates high usage of server computational expenses as the server needs to compute the root hash of file after every data update functions. The provable data possession scheme by [10] improved the computational efficiency of integrity verification by using algebraic signature for pre-processing of data. It consists of five stages of challenge and response protocol and exchanges a small, constant amount of data between data owner and the cloud storage. Furthermore, [2] also proposed a protocol based on algebraic signatures which supports the dynamic data updates for cloud storage. The dynamic data operation in the protocol is based on divide and conquer table that uses multiple tables to store dynamic data operations. The drawback or limitation of this approach is that the dynamic operation such as deletion and insertion needs rebalancing the table. Moreover, quick searching of data block is also a major flaw of this scheme [15]. From the above discussion, it can be concluded that the data integrity verification schemes for outsourced data on cloud storage are not efficient. Further, the drawbacks of schemes are that either it consumes high computational expenses and communicational overhead for data integrity metadata generation and verification. To support the data dynamic in PDP schemes they incur high computations on server because of node rebalancing in data structures. This research analysed different integrity verification protocols and will propose an efficient data integrity verification schemes for outsourced data on cloud storage supporting the dynamic (append, insert, delete, update) nature of data.

3 PRELIMINARIES

This section presents the background of the proposed Modular provable data possession scheme.It also presents the algebraic signature and linked list techniques which are used to implement proposed modular PDP.

3.1 System Model

There are three entities in the architecture of Modular PDP which are the (i) data owner who does the pre-processing on the file before uploading to the cloud storage and also does the dynamic operation such as delete, update, insert, and append. (ii) CSP that provides the storage of the data file and provides proof of the data integrity when challenged by the verifier, and (iii) verifier who could be data owner itself or any trusted third party that have the information about the secret of the file and regularly checks the integrity of the file stored on the CSP storage.

3.2 Algebraic Signature

The Algebraic signature is the short signature which is a type of hash function with algebraic properties. The main property

of the algebraic signature is that the sum of the algebraic signature of the blocks produces the same result as the signature of the sum of the corresponding blocks. The algebraic signature of the file with n blocks (f[1], f[2], f[3], … f[n]) is calculated as:

$$s(f) = \sum_{i=1}^{n} f.\gamma^{i-1} \quad (1)$$

3.3 Linked List

Linked lists in a high level perspective as being a series of nodes. A linked list is a data structure used to keep information in sequential order. The node is the basic building block of a linked list data structure. Each node contains two pieces of information: a value, which means the data which we need to store in the linked list. A link which shows the progression of data structure and store the logical address of the progressive node. The pointers are used to point to the memory location of the stored node in the database. A graphical view of linked list data structure is given below in Fig. 1 [16].

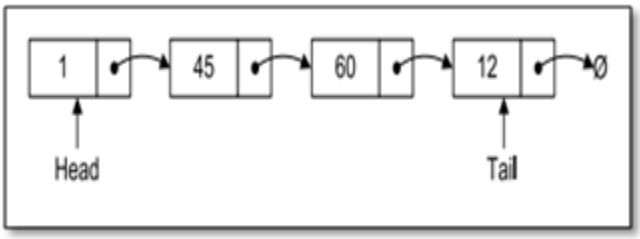


***Fig. 1**. Linked List Data Structure*

4 System Design

The design of the modular PDP scheme is based on the client server architecture in which the client is the data owner and the server is cloud service provider. The client or data owner is responsible for the pre-processing of data which is to create the data blocks and their algebraic signatures. The cloud service provider will be responsible for creating the linked list which possesses the data of logical indexes of the data blocks which are stored on cloud storage. Fig. 2shows the design of modular PDP scheme and its phases and the details of phases are provided in following section.

4.1 Pre-Processing Phase

The preprocessing of data file is done by the user of the data file in which the user preprocess the data to segment the data file into blocks to calculate the algebraic signature of each block.

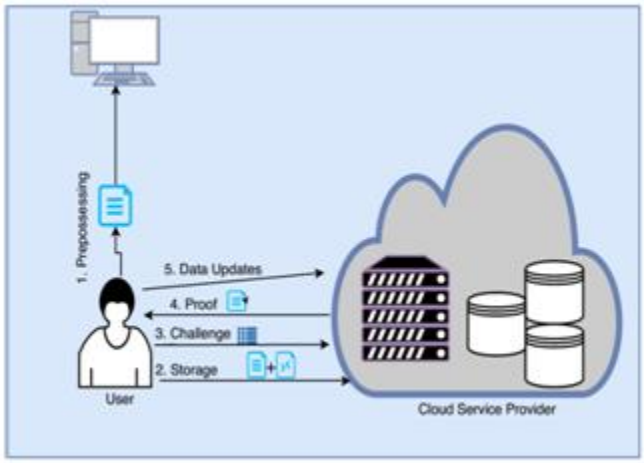


***Fig. 2**. Modular Provable Data Possession Architecture*

AlgebariacSignature(dataFile): Calculate Algebraic Signature:

i) User: Segment File F into f_n data blocks.
ii) User: Generate a secret key k.
iii) User: Create algebraic signature δ_n of each Block using secret key k.
iv) User: Output (dataFileBlocks, algebraicSignature).

UploadFile(dataFileBlocks$\mathbf{f_n}$, algebraicSignature $\boldsymbol{\delta_n}$): Upload file to cloud storage

i) User → Server: Send data blocks f_n and signature blocks δ_n to cloud storage.
ii) Server: Store the data blocks f_n and signature blocks δ_n in the cloud storage database.
iii) Server: Creates a linked list in database and store the logical index of data blocks and dynamic version number.
iv) Server → User: Notification for Success (file upload successful)

4.2 Integrity Verification

The main goal or objective of a PDP schemes is to ensure that the integrity of outsourced data is preserved. Themodular PDP follows a process to verify the integrity of outsourced data on the cloud storage. The user sends the indexes of the data blocks as challenges to the cloud server and the cloud server creates a proof based on the indexes challenged by data owner. The proof is sent to the user or verifier which verifies the proof and ensures that the data is kept safe and not modified or damaged, hence the integrity of outsourced data in cloud storage is preserved.

Challenge (indexes $\mathbf{x}$): Challenge indexes of data blocks

i) User: Chooses indexes x of data blocks whose integrity need to be verified.
ii) User → Server: Send indexes x to cloud server as challenge.

Proof : Creates proof from challenge

i) Server: Receive the indexes $x$as challenge
ii) Server: Aggregate the data file blocks $\sum f_x$ and create hash $H(\sum f_x)$.

iii) Server: Aggregate the signatures corresponding data blocks $\sum \delta_x$.
iv) Server → User: Send proof $(H(\sum f_x), \sum \delta_x)$to data owner.

VerifyProof(): User verifies proof

i) Sever → User: Receives the hash of aggregated data blocks $H(\sum f_x)$ and aggregated signature blocks $\sum \delta_x$.
ii) User: Calculates the algebraic signature of hash with the secret key k.
iii) User: Compare the signature of hash with the aggregated signature blocks.
iv) User: If equal →integrity of data is preserved.
v) User: If unequal →data integrity might be compromised.

4.3 Data Dynamic Operation

The designed modular provable data possession can provide the data dynamic operation such as append, delete, insert, and update. The update operation follows a process in which the user provides the index of the file block that needs modification. Once the user selects the index and sends the update block with data signature for that index then the server updates the linked list and also uploads the file block along with the algebraic signature.

Update(DataBlock, AlgebaricSignature, Index): Update data block

i) User: Creates the data blocks f and Algebraic Signature δ.
ii) User: Chooses the index i for update block
iii) User → Server: Sends the data block f, Algebraic Signature δ and index i.
iv) Server: Receive the data block f, algebraic signature δ and index i.
v) Server: Access the data block and signature, which need to be updated.
vi) Server: Updates the data block f_i
vii) Server: Update algebraic signature δ_i
viii) Server: Update the linked list
ix) Server → User: Notification of Success – Data updated

Delete(index): delete data block

i) User: Chooses the index i for delete
ii) User →Server: Sends the data block index i to delete
iii) Server: Receive data block index i
iv) Server: Search the index i in linked list
v) Server: Delete the linked list node of index i
vi) Server → User: Notification of Success – Data deleted

Append(DataBlock, AlgebaricSignature): Append data block

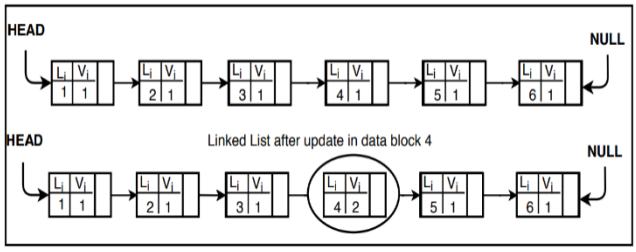
i) User: Generate the data block f and calculates Algebraic Signature δ.
ii) User → Server: Sends the data block and corresponding signature
iii) Server: Receive data block and signature of the data
iv) Server: Append the block in the text file at the last position
v) Server: Append the logical index in linked list
vi) Server → User: Notification of Success – Data Append

Insert(DataBlock, AlgebaricSignature, Index): Insert data block

i) User: Generates the Data block f and calculates Algebraic Signature δ.
ii) User: Chooses the index i where the insertion is done
iii) User → Server: Sends the data block, algebraic signature and index i
iv) Server: Receive data block, algebraic signature and index
v) Server: Store data block and algebraic signature in database
vi) Server: Insert the linked list node after index i which contains logical index of stored data block.
vii) Server → User: Notification of Success – Data inserted

5 DATA DYNAMIC OPERATIONS

The integrity verification of dynamic data at cloud storageusingdeterministic way is computational expensive because in deterministic approach after each dynamic operation on the data, the signature of whole file is recalculated. In contrast, the file probabilistic approach is segmented into blocks so if any data block is dynamically updated then only that specific block signature is required to be recalculated. This modular PDP is probabilistic in nature so this supports the dynamic property of data integrity verification. Data update is an essential feature of provable data possession schemes which supports update of data without downloading the data and calculating the signature of whole data. The proposed scheme uses linked list data structure to support data dynamics efficiently in cloud storage as it does not incur any node rebalancing for data updates. The data in



the linked list consist to two major entities which are the logical index (L_i) which stores the index of the data block and the dynamic update version number of the block (U_i) which stores the number of updates on certain data block. The CSP is responsible for generating the linked list data structure of each of the file uploaded on the cloud storage.

5.1 Data Update Operation

The data updates happen quite often on the cloud storage. For PDP the support of data updates is very important which allow the data file owner to update the data file block or just a few bits of block efficiently. In the proposed algorithm to update the i^{th} block first find the logical index of i^{th} block from linked list data structure and increase the dynamic version number by 1 of the linked list which shows an update to the data block shown in Fig. 3. Further access the data block using logical index and update the content of the data block and recalculate the algebraic signature block of that data block and finally update the signature block.

Fig. 3 *Effect of Data Update Operation on Linked List*

5.2 Data Insert Operation

Data insertion is also one of the most important feature of a PDP. Insertion of data means to insert a new data block or bits of data after the i^{th} data block of the file. The data insertion algorithm works as the user or data owner first selects the i^{th} block after which a data block need to be inserted. The user calculates the algebraic signature of new data block. The user sends the data block, corresponding calculated algebraic signature and the index insert(datablock, algebraicSignature, index) to the cloud server. The CSP first store the data block and its algebraic signature to the corresponding tables in the cloud storage. Then CSP update the linked list by inserting the logical index and version number of data block after i^{th} index. In linked list the insertion is done by just changing the nodes for the next linked list element and does not require any computationally expensive node rebalancing. As in Fig. 4 which shows the insertion of 7^{th} block after the 3^{rd} block.

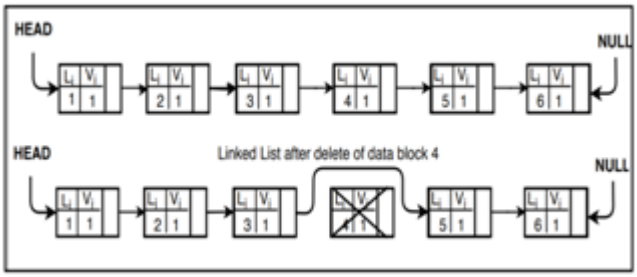


Fig. *4.Effect of Data Insert Operation on Linked List*

5.3 Data Delete Operation

The data deletion is primarily to delete some information from the data block or delete the whole block of the file. The deletion in the proposed algorithm is very efficient because the linked list does not move or rebalance the nodes. For the deletion of the data block the user sends the i^{th} block index or data bits along with the index to delete whole block or few bits of the block. The CSP finds the data block in the linked list data structure and removes the link of node from the previous node and points towards the following node shown in Fig.5.

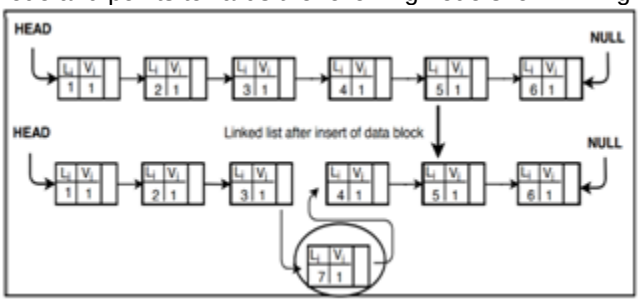


Fig. 5*. Effect of Data Delete Operation on Linked List*

5.4 Data Append Operation

The data append is basically insertion of data block at the end of the file or after the last block in the file. The append algorithm in the proposed modular PDP works as the user or data owner creates the data block and calculates its algebraic signature. After calculating the algebraic signature the user sends the data block and the corresponding algebraic signature to the CSP. The CSP insert the data block and the algebraic signature to the tables of the file. Then the CSP update the linked list data structure and adds a new node at the end of the linked list and store the index number and version number of appended data block shown in Fig. 6.

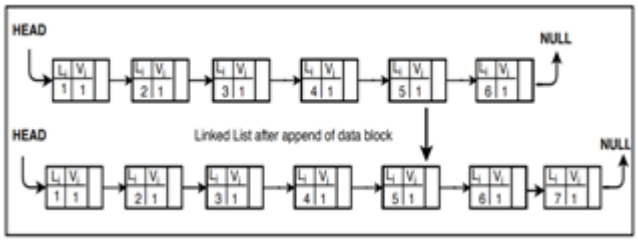


Fig. 6*. Effect of Data Append Operation on Linked List*

6 Experiment & Results

The experiment is implemented using Java programming language on a system with an Intel Core i7–2670QM at 2.20GHz, and 6.00GB RAM. The experiment is done using the text file of different sizes from 1MB to 300MB. Further the data dynamic operations also carried out using linked list and compared the results with state of the art PDP technique iTable.

6.1 Algebraic Signature Calculation

The computational expense involves the computation time required to calculate the algebraic signature of a file. The experiment for computational time is conducted on the text files of different size from 10MB to 300MB which are created using random function or random values. The data owner pre-process the data file and creates the segments of file as data blocks. For each of the data block the algebraic signature is calculated using the secret key and the computational time is recorded as shown in Fig. 6. In this figure, x-axis represents the data file size while the y-axis is the time required to calculate the algebraic signature in millisecond. As the data file size increases, the execution time to calculate the algebraic signature also increases at a linear rate which can be verified by dotted trend line which shows the linear forecast. The computational expense has the linear graph with data file size and time required to calculate the algebraic signature.

6.2 Integrity Verification

The proposed modular PDP uses probabilistic approach in which some blocks are challenged to verify the integrity of the data. This scheme use algebraic signature and linked list to verify the data with downloading a bulk of data. The experiment is conducted which uses the file sizes of 1MB to 300MB whose algebraic signature are calculated. For integrity verification the user challenge all the blocks and the aggregation of the signature blocks and the hash of the aggregation of data blocks is returned to the data owner. The data owner creates the signature of the hash by secret key and verifies the integrity of the data. The results in Fig. 7 shows the time required to check the integrity of different size of data which have only one blocks corrupted. The results shows that time required to verify the integrity of different sizes of the file is linear which can be verified by the trend line which forecast the linear growth.

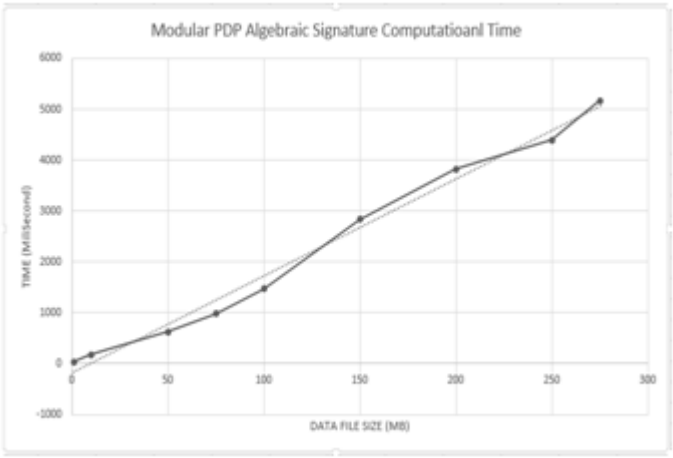


Fig. 7*. Computational Time for Data Integrity Verification*

6.3 Communicational Overhead

The communication expenses in PDP are basically the data required to send and receive from cloud storage and the data user. In the proposed modular PDP which uses the Algebraic signature and linked list to verify the integrity and perform the data dynamic operation. The communication expense for the verification of data is based on the Algebraic signature property which states that "Signature of the sum of the data blocks is equal to the sum of corresponding signature blocks". So the proposed PDP uses the 256bits data blocks for the signature and also the data blocks are hashed with SHA-256 before calculation of the signature. After the challenge send by the user the cloud storage generates the proof which is summation of the signature blocks and the summation of the hashes of the data blocks which both makes 512 bits. Hence the communication expenses for the integrity verification of the data is only 512 bits.

6.4 Dynamic Data Operation

The dynamic data operation are performed on linked list and compared with the dynamic operation of the iTable. In the simulation the total number of data blocks generated is 12000 which means the linked list and iTable have 12000 nodes or elements. The deletion of blocks is simulated over 50 to 1500 data blocks so that the nodes or elements deleted from iTable and Linked list are same as 50 to 1500. The table 1 shows the comparison of Big (O) complexity of iTable and Linked list dynamic data operations.

Table 1 *Big(0) for Linked list and iTable dynamic Operation*

Function	Linked List	iTable
Insert	O(1)	O(n)
Delete	O(1)	O(n)
Update	O(1)	O(1)
Append	O(1)	O(1)

7 COMPARISON

This section shows the comparison of modular PDP with other state of the art PDP techniques. Each column in the Table 2 shows the comparison between techniques and computational complexity for integrity verification and the dynamic data operations. First column is homomorphic based PDP proposed by Chen, 2013 which provide the data integrity verification at cost of $O(2^n)$. The exponential calculation is infeasible for huge data sizes which are stored on the cloud storage and it does not provide the data dynamic operations for data. The iTable based PDP is also based on homomorphic tags with support of data dynamic operations. The computational complexity for iTable is quite expensive in terms of integrity verification and data dynamics as well. In iTable the insert and delete operations requires high computational expenses because for each of the operation the rebalancing or re-indexing of node is required. The DCT based PDP is based on algebraic signature which is efficient way to verify the integrity of the cloud storage data. The data dynamics is supported by using DCT data structure. The DCT improves the iTable but it still have limitations when frequent data dynamic operations are required by the data owner. After large frequent data dynamic operations the DCT will resemble with iTable and will have re-indexing or re-balancing of DCT entries for each insert and delete operation. The proposed modular PDP scheme is based on algebraic signature for providing efficient integrity verification to data stored on cloud storage. The Modular PDP uses linked list to improve the limitations of existing techniques dynamic operations. The linked list does not require any rebalancing or re-indexing of the nodes and the computational complexity is only of $O(1)$ for all data dynamic operation.

Table2 *Comparison of PDP schemes*

Operation	Chen PDP [17]	iTable[18]	DCT [2]	Proposed Modular PDP
Integrity Verification	Homomorphic authenticators $O(2^n)$ Complexity	Homomorphic Verifiable Tags $O(2^n)$ Complexity	Algebraic Signature $O(N)$ Complexity	Algebraic Signature – $O(N)$ Complexity
Insert	N/A	Yes – $O(N)$	Yes – $O(\frac{N}{k})$	Yes – $O(1)$
Update	N/A	Yes – $O(1)$	Yes – $O(1)$	Yes – $O(1)$
Delete	N/A	Yes– $O(N)$	Yes – $O\left(\frac{N}{k}\right)$	Yes – $O(1)$
Append	N/A	Yes – $O(1)$	Yes – $O(1)$	Yes – $O(1)$
Limitations	Does not provide support for data dynamic operations.	High computational expense for integrity verification and data insert and delete operation.	High computational expense for data insert and delete operation.	Efficient integrity verification and dynamic operations.

8 CONCLUSION

In this paper, authors have presented the modular provable data possession scheme for cloud data storage based on algebraic signature and linked list. The algebraic signature provides the probabilistic and as well as deterministic approach

for efficient integrity verification. The communicational overhead for the integrity checking is also minimal which only 512 bits of data. Furthermore, the linked list also provides efficient data dynamic operations. On comparison with state of the art technique iTable with the linked list, the linked list performance increased by 60% for insert and delete operation.

9 FUTURE WORK

In future the implementation of the proposed PDP Scheme can be done on the real cloud environment along with the user's data in order to check the efficiency of the modular PDP. Moreover, this modular PDP scheme can be incorporated with the big data integrity verification and support data dynamic nature of cloud service data.

REFERENCES

[1] P. Jain, M. Gyanchandani, and N. Khare, "Big data privacy: a technological perspective and review," J. Big Data, vol. 3, no. 1, p. 25, 2016, doi: 10.1186/s40537-016-0059-y.

[2] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, "Dynamic remote data auditing for securing big data storage in cloud computing," Inf. Sci. (Ny)., vol. 380, pp. 101–116, 2017, doi: 10.1016/j.ins.2015.09.004.

[3] S. Singh, Y. S. Jeong, and J. H. Park, "A survey on cloud computing security: Issues, threats, and solutions," J. Netw. Comput. Appl., vol. 75, pp. 200–222, 2016, doi: 10.1016/j.jnca.2016.09.002.

[4] Y. Fan, X. Lin, G. Tan, Y. Zhang, W. Dong, and J. Lei, "One secure data integrity verification scheme for cloud storage," Futur. Gener. Comput. Syst., vol. 96, pp. 376–385, 2019, doi: 10.1016/j.future.2019.01.054.

[5] T. Wu, G. Yang, Y. Mu, R. Chen, and S. Xu, "Privacy-enhanced remote data integrity checking with updatable timestamp," Inf. Sci. (Ny)., vol. 527, pp. 210–226, 2020, doi: https://doi.org/10.1016/j.ins.2020.03.057.

[6] N. Garg and S. Bawa, "Comparative analysis of cloud data integrity auditing protocols," J. Netw. Comput. Appl., vol. 66, pp. 17–32, 2016, doi: 10.1016/j.jnca.2016.03.010.

[7] Q. Wang, S. Member, C. Wang, S. Member, and K. Ren, "Enabling Public Auditability and Data Dynamic in Cloud Computing," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847–859, 2012, doi: 10.1007/978-3-642-04444-1_22.

[8] K. Zeng, "Publicly Verifiable Remote Data Integrity," pp. 419–434, 2008.

[9] F. Zafar et al., "A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends," Comput. Secur., vol. 65, pp. 29–49, 2017, doi: 10.1016/j.cose.2016.10.006.

[10] L. Chen, S. Zhou, X. Huang, and L. Xu, "Data dynamics for remote data possession checking in cloud storage," Comput. Electr. Eng., vol. 39, no. 7, pp. 2413–2424, 2013, doi: 10.1016/j.compeleceng.2013.07.010.

[11] P. Shah and P. Prajapati, Provable Data Possession using Additive Homomorphic Encryption. Springer Singapore, 2020.

[12] Y. Yu et al., "Identity-Based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage," IEEE Trans. Inf. Forensics Secur., vol. 12, no. 4, pp. 767–778, 2017, doi: 10.1109/TIFS.2016.2615853.

[13] G. Ateniese et al., "Provable data possession at untrusted stores," Proc. 14th ACM Conf. Comput. Commun. Secur. - CCS '07, no. 1, p. 598, 2007, doi: 10.1145/1315245.1315318.

[14] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," CCS '09 Proc. 16th ACM Conf. Comput. Commun. Secur., pp. 213–222, 2009, doi: 10.1145/1653662.1653688.

[15] Y. Ping, Y. Zhan, K. Lu, and B. Wang, "Public data integrity verification scheme for secure cloud storage," Inf., vol. 11, no. 9, pp. 1–16, 2020, doi: 10.3390/INFO11090409.

[16] D. Structures, "Annotated Reference with Examples."

[17] L. Chen, "Using algebraic signatures to check data possession in cloud storage," Futur. Gener. Comput. Syst., vol. 29, no. 7, pp. 1709–1715, 2013, doi: 10.1016/j.future.2012.01.004.

[18] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 9, pp. 1717–1726, 2013, doi: 10.1109/TPDS.2012.278.