International Journal of Engineering & Technology, 7 (2.24) (2018) 500-503

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Data Integrity Verification Using MPT (Merkle Patricia Tree) in Cloud Computing

Subasri Mathiyalahan[*], Shobana Manivannan, Mahalakshmi Nagasundaram, R Ezhilarasie

School of Computing, SASTRA Deemed to be University, Thanjavur-613401
**Corresponding Author E-mail: subasrimathiyalahan95@gmail.com*

Abstract

Data integrity of outsourced data is main problem in CSP (cloud service provider). Space overhead and computation complexity are very high issue in recent PDP(Provable Data Possession) verification schemes. To overcome such issues MPDP (Mobile Provable Data Possession) schemes using hash tree data structure and Boneh-Lynn-Snacham short signature scheme have been used over decade. Data dynamics is well supported in MPDP scheme via block less verification, dynamic data operations, stateless verification, and verification out sourcing. But still there are some operations which can be performed much more efficiently in some other way than that of the two methods prescribed above. Operations in particular, data modification operations like insertion and deletion operations is somewhat difficult or in other words time consuming in hash tree data structure. In this paper, we have deployed an improved hash tree structure called MPT (Merkle Patricia Tree) for integrity checking.MPT is combination of MHT (Merkle Hash Tree) and patricia tree where each node consists of key-value pairs. As of now, MPT has been used only in block chain technology for providing authentication of transactions through Ethereum.

Keywords: *Data Integrity, Provable Data Possession (PDP), Patricia Tree, Merkle Hash Tree, Merkle Patricia Tree, Block chain.*

1. Introduction

Cloud Computing is one of the fastest emerging technologies and widely used for discrete service like servers, storage, software development platforms as Platform as a Service (PaaS), Software as a Service (SaaS) and Infrastructure as a Service (IaaS). The major concern in cloud computing is the vulnerability of the data,since the data and resources are shared among the many users and there is a higher chance of misuse or lose of data [7].Data integrity process invokes the approach of assurance on data and ensures that data is original, authentic and shielded from unauthorized user alteration. It helps in identifying whether the data are damaged by service owner's hardware fault or by forgetful operation or by an adversary malignant attack [2].The elements that are inferred from the integrity analysis are Stateless verification, Public Verification, Support dynamic operation, Batch Auditing. In stateless verification phase, anyone can do the verification of data successfully. Public Verification**,** the user can invoke a separate third party auditor to check the storage available on the cloud. The data that is outsourced is accessed as well as updated by the user in Support dynamic operation. Batch Auditing involves multiple delegated auditing functions from various clients and it can be achieved synchronously by the (Third Party Auditor) TPA in a privacy-preserving manner [5].
Over the years, various data integrity scheme like Provable Data Possession(PDP) , POR(Proof Of Retrievability) and MPDP(Mobile Provable Data Possession)have been proposed for protecting outsourced data. The Provable Data Possession (PDP) scheme plays amain role in Cloud Computing to design data oriented security architecture. Without PDP scheme, customer's cannot authenticate the server whether the processed data is real data or not[3].POR (Proof Of Retrievability) is used for handling large files and users are allowed to retrieve the file without having any computation overhead along with guaranteed QoS. In MPDP, Boneh-Lynn-Snacham, Merkle hash tree concept are the 2 methods used to enhance efficiency and data integrity [4]. In MHT data structure, data blocks can be arranged in flexible manner and has a minimum transmission cost, whereas easy verification of data corruption is provided by bilinear mapping and BLS scheme.

2. Preliminaries

2.1 System Architecture

There are three entities viz, 1) data owner, owns the data which is stored in the cloud 2) CSP (Cloud Service Provider),provides the storage service which is utilized by the customers 3) TTP (Trusted Third party),responsible for assuring the integrity of the data requested by the owner.

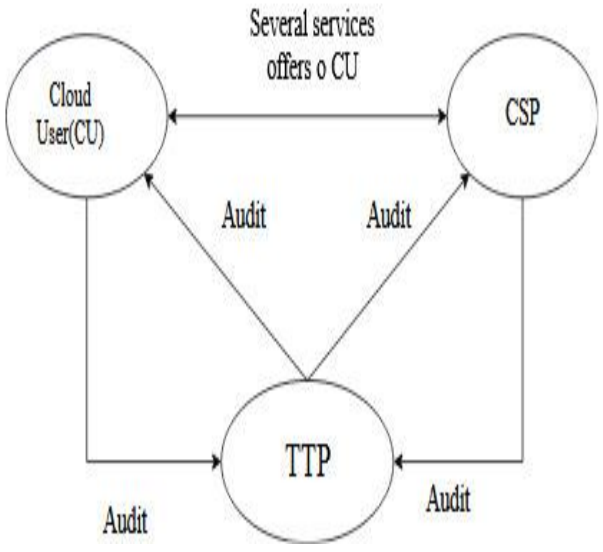

Fig. 1: Scheme Architecture

2.2 Methodology

The integrity of data in the cloud can't be guaranteed effectively because of the following reason: 1) Users lose the confidentiality of data. 2) Conventional cryptographic analyzing method can't ensure the assurance of data security. Recently, MHT is a prevailing method for integrity checking in any field especially in cloud based data storage. MHT provides the some data security, but the insertion and deletion operation are difficult to perform. Our Paper depicts a better data structure, MPT(Merkle Patricia Tree) than the existing one to tackle the unforeseen challenges. Performance can be effectively increased by making use of MPT for insertion and deletion operations.MPT is the fusion of Merkle tree concept and Patricia Tree. The key features provided by MPT includes: i)storage in the form of key value pair, ii)Lightweight expanded node, merkle proof, iii) quick state of rolling mechanism. iv)Rapid calculation of hash functioning of data.

2.2.1 Patricia Tree

Patricia tree is a slight modification of prefix tree. In the case of prefix tree, location of the prefix node is decided by the details of its key i.e., the data is encoded in path of the node from root node. But in Patricia tree, the data value is stored in a single node unless a new data with the same prefix as the stored one is inserted.

2.2.2Merkle Patricia Tree (MPT)

MPT is the Combination of MHT(Merkle Hash Tree) and Patricia Tree. The time Complexity of the MPT for insert, delete and search is $O(\log(n))$.The advantage of using the MPT is to reduce the time taken for modifying the data similar to insertion and deletion operations. In this data structure, every node that is a single child is combined with its parent.

2.3 Pseudocode for Operation of MPT

```
Algorithm MPTree(File, MPT)
    returns hash value of the root of Merkle Patricia Tree
    Persistent:  File - A text file that is to be stored and verified for data integrity.
                 MPT - A tree structure initially empty.
    wordfile<- Read the input File word by word and store it in a new file.
    for each word in wordfile:
        MPT <- MPT.insert(word)
    Root <- MPT.root
    filehash<- MerkleHashRoot(Root)
    returnfilehash
Function insert(word)
    returns MPT with the word inserted in it.
    Persistent:  word - word to be inserted into the MPT structure.
                 MPT - Tree structure where the word is to be inserted.
    if MPT is empty then
    create a new node and store the word in the node as its value.
    else
    for each node in MPT:
        compare the word and node.value
        if both are equal then
            return MPT
        elseifnode.value is a prefix of the word then
            create a child for the node with value (word-node.value)
        elseif the word is a prefix for node.value then
            split the node at its first difference with the word.
        else
            create a new node with a value of word.
    return MPT

Function MerkleHashRoot(root)
    returns hash of the file
returnMerkleHash(root)

Function MerkleHash(root)
    returns hash of the Merkle Patricia Tree
    Perisistent:  root - The root node of Merkle Patricia Tree.
    ifnode.children is empty then
        return Hash(node.value)
    children_list<- node.children
    temp<- Hash(node.value)
    for all child in children_list:
        temp<- temp + MerkleHash(child)
    return Hash(temp)
```

According to the algorithm an empty tree structure with root node as null is created. When the file containing the data to be stored is given as input to this algorithm, it reads the file word by word and each of these words will be inserted into the tree immediately after it is read. So the first word will be inserted into an empty tree thereafter the words will be compared with all the nodes' data to check for similarity between the two and if there is partial difference then the point of first difference is saved for further operations. If the new word is same as the node's word then no new node is created and added to the tree. If the new word forms a prefix for the node's data then the node is divided into two nodes, one as parent with the common prefix value of the two words and the other as the child of the node with the remaining word. If the node's word is entirely a prefix for the new word then the new word with the removal of the prefix word from it will be inserted as a child for the node. If the new word is entirely different from all of the existing words in the tree then a new node is created and added to the tree. These words stored in the tree nodes are considered as the keys. When the file is read completely and the tree is formed, the root of the tree is passed to a function for calculating the hash values for each of the node and storing it as the values for the keys stored in the tree. In the function, the tree is traversed from the children nodes to the root consolidating the hash values of the children towards each of its parent. At the end of this algorithm, the hash of the entire tree will be obtained as a result of consolidated hash value calculation.

3. Results and Discussion

MHT is the complete binary tree format, but in our paper, we have employed MPT data structure which can have a maximum of 26 children for each alphabet. The approach employed in this work follows Merkle tree concept for deriving calculated hash values and to have key value pairs Patricia Tree concept is used. Each node in tree contains key and a hash value. Here, the key in each node represents either a prefix value or the entire data in case of single child and the value's data is consolidated hash of its own key and its children's value if exist.

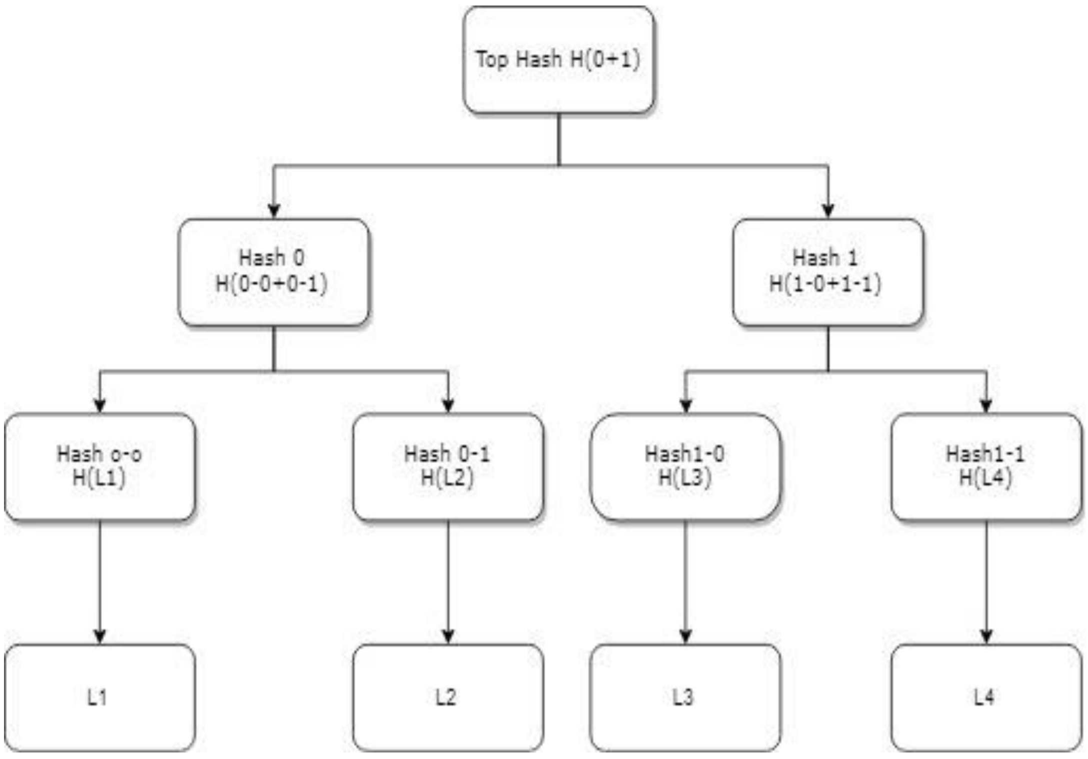

Fig. 2: MHT data structure

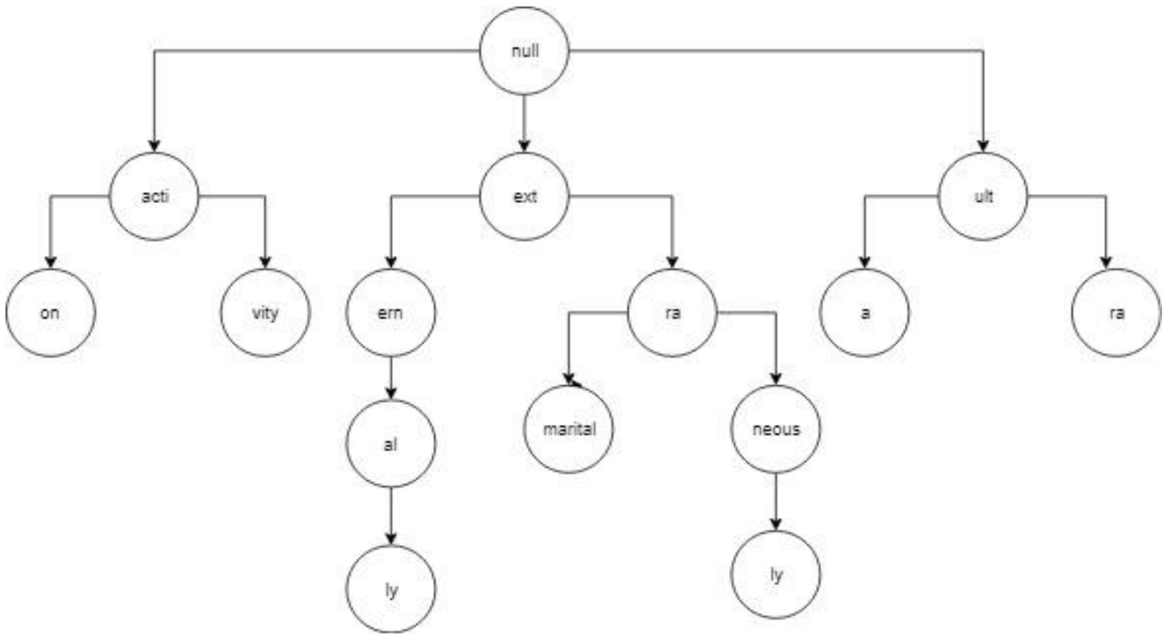

Fig. 3: MPT data structure

In fig 2.,it is the patricia tree data structure devoid of hash values whereas in case of MPT each node will have its corresponding hash values calculated by MHT format along with its key data value. These kinds of data structure reduce the extra disk access and also insertion and deletion operations are easy to perform. The root node of the MPT data structure is always null. In fig 3 some words are arranged in prefix order. Those words are action, activity, extern, extra, external, externally, extraneous, ultra, ulta. Hash values are calculated for each and every node after storing the word and traversed in the reverse order from the leaf to the root. If there is a modification of a single letter in any of the node, it is very easy to update but at the same time the hash value of particular node will be changed which ultimately causes the hash value of the root node to be changed. For example if we take words like 'action' and 'activity' with a common prefix 'acti' then the prefix will for a node 'acti', it will be stored as the first node on its path from the root followed by its two distinct suffixes ‘on’ and ‘vity’ as its children. So for calculating hash values the prefix node must take into consideration all its children nodes. Table 1 Shows hash value of two subtrees with prefixes ‘acti’ and ‘ult’ along with its children and shows hash value of modified data file(act-atc) which affects not only its individual hash value but also the root has but it will not affect others subtrees.

Table 1: Initial hash value and hash value after modification of data

Root 60bd0a02ca4dace4ac07d22e8b0d09bb	**Root c89fb30beddb290c0035a7aa6a7ffcafd**
act 51fcfa78c0646f287587a8123792f0ff	atc db26ee047a4c86fbd2fba73503feccb6
on ed2b5c0139cec8ad2873829dc1117d50	on ed2b5c0139cec8ad2873829dc1117d50
vity 0af5e227cb909796dfb03430f1fcd6f6	vity 0af5e227cb909796dfb03430f1fcd6f6
ult a0ad7803047afc61cab73de5d41ad539	ult a0ad7803047afc61cab73de5d41ad539
a 0cc175b9c0f1b6a831c399e269772661	a 0cc175b9c0f1b6a831c399e269772661
ra db26ee047a4c86fbd2fba73503feccb6	ra db26ee047a4c86fbd2fba73503feccb6

Table 2: Shows the relationship between MPT and MHT efficiency

	Time taken for insert Operation	Time taken for delete operation	Time taken for update operation	Time taken for search operation
MHT	High	High	High	High
MPT	Low	Low	Low	Low

4. Conclusion

Security of the cloud computing is still challenging task. Many algorithm and data structures are proposed for this integrity verification.MHT data structure is one of the best integrity checking algorithms. But the insertion and deletion operations are somewhat difficult to be performed in MHT. In this paper, we analyze the functionality of MPT data structure (integrated MHT) which functions similar to MHT but the difference is insertion and deletion operations can be performed much easier when compared to MHT. Both the MPT and MHT are depends on Trusted Third Party (TTP).In future work we will expand our scheme to untrusted TTP.

References

[1] M. S. Niaz and G. Saake, “Merkle hash tree based techniques for data integrity of outsourced data,” CEUR Workshop Proc., vol. 1366, pp. 66–71, 2015.

[2] A. Shoufan, N. Huber, and H. Gregor Molter, “A novel cryptoprocessor architecture for chained Merkle signature scheme,” Microprocess. Microsyst., vol. 35, no. 1, pp. 34–47, 2011.

[3] V. Verma, “An Efficient Signcryption Algorithm using Bilinear Mapping,” Computing for Sustainable Global Development (INDIACom), 3rd International Conference,pp. 680–682, 2016.

[4] C. Lin, Z. Shen, Q. Chen, and F. T. Sheldon, “A data integrity verification scheme in mobile cloud computing,” J. Netw. Comput. Appl., vol. 77, pp. 146–151, 2017.

[5] N. Garg and S. Bawa, “RITS-MHT: Relative indexed and time stamped Merkle hash tree based data auditing protocol for cloud computing,” J. Netw. Comput. Appl., vol. 84, no. January, pp. 1–13, 2017.

[6] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” Proc. 4th Int. Conf. Secur. Priv. Commun. netowrks - Secur., 2008.

[7] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” J. Cryptol., vol. 17, no. 4, pp. 297–319, 2004.

[8] V. Kher and Y. Kim, “Building trust in storage outsourcing: Secure accounting of utility storage,” Proc. IEEE Symp. Reliab. Distrib. Syst., pp. 55–65, 2007.

[9] A. . Juels and B. S. . Kaliski Jr., “Pors: Proofs of retrievability for large files,” Proc. ACM Conf. Comput. Commun. Secur., pp. 584–597, 2007.

[10] G. Xu, Z. Sun, C. Yan, and Y. Gan, “A rapid detection algorithm of corrupted data in cloud storage,” J. Parallel Distrib. Comput., vol. 111, pp. 115–125, 2018.

[11] E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and integrity in outsourced databases,” ACM Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.

[12] J. Zhang, Z. Zhang, and H. Guo, “Mobile Cloud Computing,” vol. 16, no. 11, pp. 3222–3235, 2017.

[13] J. Xu and E. Chang, “Towards Efficient Provable Data Possession,” Organization, pp. 1–16, 2007.

[14] R. Arora and A. Parashar, “Secure User Data in Cloud Computing Using Encryption Algorithms,” Int. J. Eng. Res. Appl., vol. 3, no. 4, pp. 1922–1926, 2013.

[15] R. Popa, J. Lorch, and D. Molnar, “Enabling security in cloud storage SLAs with CloudProof,” Proc. USENIX, pp. 355–368, 2011.

[16] J. Yuan and S. Yu, “Public Integrity Auditing for Dynamic Data Sharing With Multiuser Modification,” vol. 10, no. 8, pp. 1717–1726, 2015.

[17] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing Computer Security – ESORICS,” IEEE Trans. Parallel Distrib. Syst., vol. 5789, no. 5, pp. 355–370, 2009.

[18] P. Ora and C. Scienc, “Data Security and Integrity in Cloud Computing Based On RSA Partial Homomorphic and MD5 Cryptography,” International Conference on Computer, Communication and Control (IC4), 2015.

[19] C. Sarika and R. M. Jasmine, “An Outsourced Proof of Retrievability for Dynamic Data Operation in Cloud Abstract,” International Journal for Research in Science Engineering and Technology, vol. 3, no. 1, pp. 19–22, 2016.

[20] G. Ateniese et al., “Provable data possession at untrusted stores,” Proc. 14th ACM Conf. Comput. Commun. Secur. CCS, no. 1, pp. 598, 2007.

[21] SHUBHANSHU GUPTA, S. KOLANGIAMMAL, T.PADMAPRIYA, “Smart Curtain Using Internet Of Things” International Innovative Research Journal of Engineering and Technology, Vol. 2, Special Issue, pp. 4-8.

[22] M. Rajesh, Manikanthan, “ANNOYED REALM OUTLOOK TAXONOMY USING TWIN TRANSFER LEARNING”, International Journal of Pure and Applied Mathematics, ISSN NO: 1314-3395, Vol-116, No. 21, Oct 2017.