Enhancing Data Dynamics and Storage Security for Cloud Computing using Merkle Hash Tree and AES Algorithms

Poonam M. Pardeshi
Department of Computer Engineering
Pune University, Maharashtra,
India

Deepali R. Borade
Professor, Department of Computer Engineering
Pune University, Maharashtra
India.

ABSTRACT

With increase in the number of cloud users and the amount of sensitive data on cloud, security of cloud has become more important. Massively scalable data centers are provided by the cloud which can be accessed from anywhere and at anytime. Cloud computing allows users to store data and access it on demand thereby utilizing fewer resources in client system. However many malicious activities in cloud have accompanied the growth of cloud users. One of the greatest security challenges is storage security in cloud. It must make possible for users to store data without worrying about the need to verify its integrity. Thus, enabling public auditing for cloud storage is of critical importance so that users can restore to a third-party auditor (TPA) to check the integrity of outsourced data and be worry free. In this paper we make use of Public-Auditing. We propose a way in which the Merkle Hash Tree (MHT) used in a method called RSASS, is made dynamic by using the concept of relative index to compute the index of leaf node quickly and a dynamic operation scheme based on this tree structure for cloud storage. Also, instead of using RSA algorithm, we have made use of AES algorithm because AES requires less encryption-decryption time as well as less buffer space as compared to RSA algorithm. We thus propose a simple data protection model where data is encrypted using Advanced Encryption Standard (AES) before it is launched in the cloud, thus ensuring data-confidentiality and security.

Keywords

Advanced Encryption Standard, Cloud Computing, Data Integrity, Dynamics Data Operations, Merkle Hash Tree, Public Auditability, Storage Security.

1. INTRODUCTION

Cloud storage is an important service provided by cloud computing which makes it possible for data owners to move their data from local computers to the cloud. Moving the data on the cloud provides users with the facility of remotely storing data thereby freeing up local disc space and enjoy on demand availability of data. Both Amazon Simple Storage Service (S3) and Amazon Elastic Cloud (EC2) are well known examples of this.

It has made possible for the users to subscribe high quality services of data and software which reside solely on the remote servers and enjoy the on demand provision of the services. As a huge amount of storage space and customizable computing resources are provided by internet based online services, the shift to online storage has greatly contributed in eliminating the responsibility of local machines in maintenance of data.

Though cloud computing brings many benefits, it also puts forth many great challenges privacy protection and data security fields [11, 12].The remote data at untrusted stores lack data integrity. Owners would worry that the data stored at the cloud could be lost. Although the cloud infrastructures are more powerful than the personal computing devices, there still lie some external threats to the cloud storage. For example, the Cloud Service Provider (CSP) may discard some of the less frequently used data to save the storage space and hence increase profit margins. Some of the CSPs may also attempt to hide the data loss to maintain the reputation. Therefore, although outsourcing data into the cloud seems to be attractive economically, the data integrity and availability factor may impede its adoption by users. The data owners therefore need to be convinced that the data are correctly stored in the cloud. In order to achieve data integrity and availability and enforce the quality of cloud storage service, efficient methods that provide on-demand data correctness verification on behalf of the users are used. The data integrity problem is solved by many systems. All of them fall under public auditability and private auditability. In public auditing, the client delegates the authority of verifying the integrity of data at the server to the Third-Party Auditor (TPA) [3]. The TPA is an entity that monitors the integrity of data stored at the untrusted server on behalf of the client. Various algorithms can be used by the TPA in order to check the integrity. The Storage Security method is used for auditing the data stored at remote server [14].

Along with storing the data on the cloud server, the users may need to perform dynamic operations such as insert, delete, update and modify. In order to provide support for data dynamics, this paper constructs a Dynamic Merkle Hash Tree (DMHT) [10] structure with relative index of leaf node.

According to a performance evaluation, if we go from AES-128 to 192 bits key, the power and time consumption increases by 8% and 256 bits key causes an increase of 16 % [15]. So we propose use of industry-standard high grade Advanced Encryption Standard (AES) symmetric encryption algorithm with key length of 128-bits for this purpose.

The paper is organized as follows, section 2 describes background details, section 3 explains literature survey, section 4 describes proposed system, section 5 describes conclusion of the paper.

2. BACKGROUND THEORY

The data integrity problem is solved by many systems. Some make use of two-party auditing process [3] while some use third-party auditing. In two-party auditing, the client itself sends the challenge to the server and the server is supposed to respond to it with a proof to prove that it contains the data in integrated manner. In third-party auditing, however, the client delegates the right of auditing the data at the server to a third-party called as the Third-Party Auditor.

2.1 Third Party Auditor (TPA):

The TPA performs auditing on behalf of the Client. The introduction of the TPA reduces the overhead of the client. The client no longer needs to verify the integrity of the data at the server on its own.

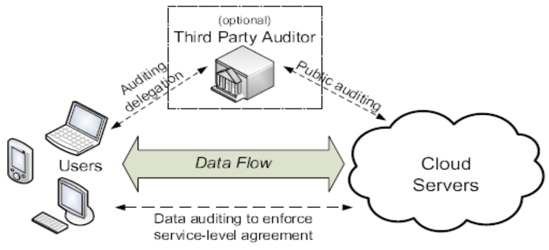

Fig.1: Cloud Data Storage Architecture [1]

Fig.1 shows three entities viz. the client system, the cloud service provider and the Third Party Auditor. The client generates the data and sends the file data to the remote cloud server. The TPA analyses the integrity of the stored file in the server and report it to the client about the status of the file data. If the file is affected, any intrusion or attack is notified to the client.

2.2 Merkle Hash Tree (MHT):

A Merkle Hash Tree is a well-studied structure used for authentication purpose [7], which is intended to prove efficiently that a set of elements are unaltered and undamaged. It is used for decreasing the server computation time [9]. It is used by cryptographic methods to authenticate the file blocks. The tree is constructed as a binary tree where the leaf nodes are the hashes of the authentic data values i.e. the original file blocks. The idea used in this is to break the file up into a number of small pieces, apply hash to these pieces and the combine iteratively and rehash the resulting hashes in a tree-like fashion until we get a tree with a single 'root hash'. The MHT is generated by the client and is stored at both the client and the server side. An example of the MHT structure is as shown Fig 2. Among it, ha=h(h(m_1)||h(m_2)) and h_b=h(h(m_3)||h(m_4)), where h is a secure one-way hash function.

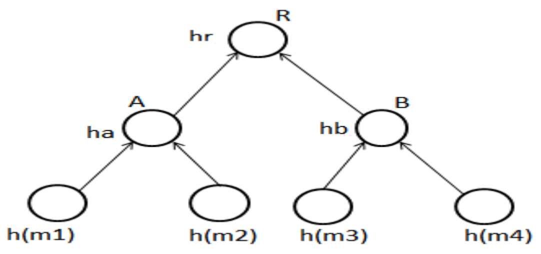

Fig.2. Merkle Hash Tree (MHT)

The authentication of the file blocks is done by the client by requesting the server to send block related information for generating the tree. This information is called as the Auxiliary Authentication Information (AAI). For example, consider the MHT in Fig.3. The verifier with the authentic root hr requests for {m_2,m_7} and requires the authentication of the received blocks. The AAI Ω_2=<h (m_1), h_d> and is Ω_7=<h (m_8), h_e> are provided by the prover to the verifier. The verifier can now verify m_2 and m_7 by computing h(m_2), h(m_7), h_d=h((h(m_3)||h(m_4)), h_e=h(h(m_5)||h(m_6)), ha=h(h_c||h_d), h_b=h(h_e||h_f) and h_r=h(h_a||h_b).

2.3 Cryptography:

Cryptography is basically used for the protection of data. Using cryptographic methods, the data is converted into secret form so that it cannot be read and understood by anyone who has no authority to do so. It can be applied using various following methods.

1.3.1Symmetric Key Cryptography:

In Symmetric Key Cryptography, both the sender and the receiver make use of the same key. In this, same key is used for encryption as well as decryption. The AES algorithm used in this paper falls under this category.

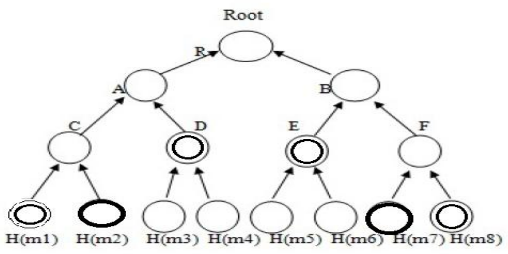

Fig.3.Authentication of data elements using MHT

1.3.2Public Key Cryptography:

In Public Key Cryptography, two different keys are used for encryption and decryption. The public key is used for encrypting the data and it can be distributed freely. However, the private key, used for decrypting the data, remains only with the receiver. Well known algorithms such as Rivest, Shamir and Adleman (RSA) and Diffie Hellman fall under this cryptographic method.

1.3.3Hash Functions*:*

These are different from SKC and PKC. They have no key at all and are also called one-way encryption. Hash functions are mainly used to ensure that a file has remained unchanged. In this paper, we make use of Secure Hash Algorithm-1 (SHA1) for applying hash to the leaf nodes of the MHT.

2.4 Cryptography Goals:

There are five main goals of cryptography. Every security system must provide a bundle of security functions that can assure the secrecy of the system. These functions are usually referred to as the goals of the security system. These goals can be listed under the following five main categories (Earle, 2005):

- **Authentication:**

The process of proving one's identity. This means that before sending and receiving data using the system, the receiver and sender identity should be verified.

- **Privacy/confidentiality:**

This goal is meant for ensuring that no one can read the message except the intended receiver. Usually this function is how most people identify a secure system. It means that only the authenticated people are able to interpret the message content and no one else.

- **Integrity:**

Assuring that the data of the client is not modified or changed i.e. the data is in its original.

- **Non-repudiation:**

A mechanism to prove that the sender really sent this message. Means that neither the sender nor the receiver can falsely deny that they have sent a certain message.

- **Service Reliability and Availability:**

Since secure systems usually get attacked by intruders, which may affect their availability and type of service to their users. Such systems provide a way to grant their users the quality of service they expect.

From the above mentioned goals, we try to achieve first three goals. We provide a mechanism that tries to achieve maximum security for cloud data by leveraging the capabilities of cryptography.

2. LITERATURE REVIEW

Most of the work in storage security in cloud computing is concerned with the integrity of the data at the remote server. Deswarte et al. in [1] makes use of RSA based hash function to verify the file stored at the remote server. Using their scheme, the client can perform multiple challenges using the same metadata.

Disadvantage: The limitation of this scheme lies in the computational complexity at the server which must exponentiate all the blocks in the file.

Miller and Schwarz in [2] proposed a technique for ensuring the data stored remotely across multiple sites. Algebraic signature was used for this purpose. This scheme makes use of a function to fingerprint the file block and verifies if the signature of the parity block is same as the signature of block.

Disadvantages: 1) The computation complexity at the server and the client side takes place at the cost of linear combination of file blocks. 2) Also, the security of this scheme remains unclear.

Public Auditing was first considered by Ateniese et al. [3] for ensuring possession of files on untrusted storages. The scheme utilizes RSA based homomorphic tags for auditing outsourced data thus achieving public auditing. In this protocol, it is considered that clients need to verify that the server has retained file data without retrieving the data from the server and without having the server access the entire file.

The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The Provable Data Possession [PDP] model for remote data checking supports large data sets in widely-distributed storage systems. It is provably-secure scheme for remote data checking.

Disadvantages: 1) This method imposes, on client, an overhead of generating metadata. 2) Does not support Dynamic Auditing. 3) Requires more than 1kilo-byte of data for a single verification. 4) It makes use of only two-party auditing Protocol, which is not efficient because neither the client nor the cloud service provider can give assurance to provide balance auditing.

Juels and Kalisiki [4] propose a scheme called “Proofs of Retrievability” (POR) which focuses on static archival of large data files. It makes use of spot checking and error correcting codes to ensure data possession and retrievability. Some special blocks called as “sentinels” are randomly embedded into the file F for detection and then the encryption of the file is carried out in order to protect the position of these sentinel blocks. Unlike PDP scheme the POR scheme cannot be used for public databases. In other words, POR scheme can only be used for confidential data.

Disadvantages are: 1) Number of queries clients used is fixed priori. 2) Introduction of sentinel nodes prevents dynamic updation. 3) Each file need to be pre-processed prior to storage at the server. 4) The scheme can only be used for confidential data and not for public databases. 5) Public Auditability is not supported.

Scalable and Efficient Provable Data Possession (S-PDP and E-PDP) protocols contribute to the work of Ateniese et al. [5]. In this paper, a dynamic version of prior PDP scheme relies only on efficient symmetric-key operations in both setup and verification phases. It provides better performance on client side, requires much less storage space and uses less bandwidth (size of challenges and responses is very small, less than a single data block). This scheme is more efficient than POR as it requires no bulk encryption of outsourced data.

Disadvantages: 1) The system imposes a priori bound on the number of queries which can be answered. 2) This concept is applicable only for static data blocks and not dynamic data operations, i.e., it only allow basic block operations with limited functionality. 3) Block insertions cannot be supported and so it is a partially dynamic scheme not fully dynamic. 4) Since the scheme is based on symmetric key cryptography, it is unsuitable for public verification.

The scheme proposed by C.Erway el at [6] is a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers. This scheme requires the server to send the linear combination of data blocks to the auditor to auditor for verification. This method makes use of Third Party Auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in cloud. It also supports data dynamics via the most general forms of data operation, such as block modification, insertion and deletion.

Disadvantages: 1) The main disadvantage of this scheme is that this scheme may leak the data content to the auditor because it requires the server to send linear combinations of data blocks to the auditor for verification. 2) The efficiency of this scheme is not clear.

Table 1 describes the comparison of existing literature reviewed system with proposed system.

Table 1: Comparison of different systems

Ref.No. / Attributes	[3]	[4]	[5]	[7]	Proposed System
Privacy Preserving	N	Y	N	N	Y
Unbound no. of queries	Y	N	N	Y	Y
Data Dynamics	N	N	Y	Y	Y
Public Verifiability	N	N	N	Y	Y
Blockless Verification	Y	Y	Y	Y	Y
Use of TPA	N	N	N	Y	Y

3. PROPOSED SYSTEM

Problem Statement: Auditing the data at the cloud server is necessary to prevent data integrity and assure the client safety of his data. The data stored is altered by the client as per his wish. However, most of the systems concentrate on provide support auditing only the static data [4, 6, 7]. So dynamic data operation is another problem of which clients are concerned about. Also it is not feasible to download entire data file from server for integrity checking.

The Merkle Hash Tree is used for authentication. In this paper, Dynamic Merkle Hash Tree is used to support dynamic data operations and provide integrity verification.

In this paper, SHA1 algorithm is used for hashing purpose and AES for is used for encryption.

Design

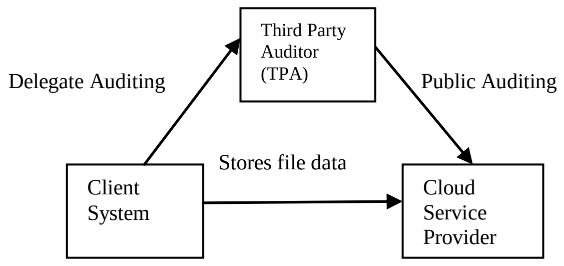

Fig.4: General Data Flow Architecture

Fig.4 Represents general data flow architecture. The three network entities used it are as follows:

- *Client (users)*: an entity that stores data files on the cloud server and relies on it for storage and maintenance of the data.
- *Cloud Server (CS)*: an entity that provides significant storage space and computation resources to store and maintain the client's data. It is managed by Cloud Service Provider (CSP).
- *Third Party Auditor (TPA)*: a trusted entity which has expertise and capabilities that client does not possess. It analyses the integrity of the stored file in the server using the RSA based signature generation algorithm

1) Algorithms

keyGen (key generation):

For the generation of the key, a random string generation algorithm is used to create a unique key. The key so generated is then encrypted by using Blowfish Algorithm [16] for security purpose. It can be used as a replacement for the DES Algorithm. It takes variable key length ranging from 32 bits to 448 bits and the default size is 128 bits. Blowfish has variants of 14 rounds or less.

In this paper, this algorithm is used to encrypt the key which has to be passed in the AES algorithm. This is done to provide extra security.

Advanced Encryption Standard (AES):
AES is a block cipher. The algorithm supports a variety of key sizes as 128,192 or 256. The default size is 256 bits. The encryption of data blocks is done in 10, 12 and 14 rounds depending on the size of the key used. It provides fast and flexible encryption and can be easily implemented on various platforms.

In this paper, AES-128 is used and so encryption is done in 10 rounds. This algorithm is used for both encryption and decryption. For encryption, it takes data blocks and the secret key as the input and outputs the encrypted data blocks. For decryption, encrypted data blocks and key are given as inputs and original file blocks are the output.

Why AES?

- AES has speedy key setup time and a good key agility.
- It is suitable for restricted-space environments as the memory requirement for its implementation is less.
- It makes efficient use of resources due to its inherent parallelism which results in a very good software performance.
- It does not have any serious weak keys.
- Any block size and key sizes are supported by AES that are multiples of 32 (greater than 128-bits)
- No linear and differential cryptanalysis attacks have yet been proven on AES.

ProofGen:
The proof generation algorithm (proofGen) generates proof for the challenge sent by the verifier. This algorithm takes the metadata and AAI as input and generates a proof P in output. The server (prover) generates the tag block T, data block M and Auxiliary Authenticate Information AAI which are necessary for the client to generate the MHT. The algorithm outputs proof as

$$P = \{T, M, \{H(m_i), \Omega_i\}_{s1 \leq i \leq sc}, sig_{sk}(H(R))\}$$

verProof:

Verify proof (verProof) algorithm is used by the client to verify the proof sent by the server. A block diagram of this process is shown in Fig.5. The proof generated by the client is given as input to this algorithm. The proof is verified in two steps. In the first step, the TPA

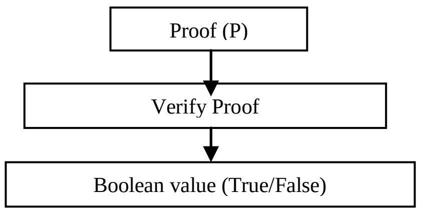

Fig.5: Block diagram of proof verification process.

or client validates the proof by generating tree with the help of AAI. If the step one is TRUE, the verifier further validates the blocks, otherwise it FALSE is emitted. This algorithm, thus outputs the Boolean value TRUE/FALSE and indicates whether the proof is valid or not.

After the proof is generated, if the output is false, i.e. the data is not in the integrated state, we check the file further to determine exactly which block of the file is corrupted or modified. This is done by comparing each block of the MHT generated by using AAI (which was generated by server at the time of proof generation) to the one which was generated during data storage.

2) Construction of DMHT:

Each node of a MHT contains a hash value; on the other hand, each node of DMHT carries two auxiliary information viz. hash and relative index. Relative index is an extra data field carried by the DMHT. It indicates the total number of leaf nodes in the subtree of a node. Therefore, if there is a node w with left child a and right child b, the auxiliary information carried by a will be (h_a,n_a) and that carried by b will be (h_b,n_b). The internal node w will have the index as $n_w=n_a+n_b$ and the hash of it is updated as $h_w=h(h_a||n_a||h_b||n_b)$.

The Fig.6 shows an example of the DMHT with relative index. The AAI Ω_5 =<(h(m$_6$),1,r),(h(m$_7$),1,r),(h$_f$,2,r),(h$_a$,4,l)> where l indicates the left sibling of the node and r indicates the right sibling.

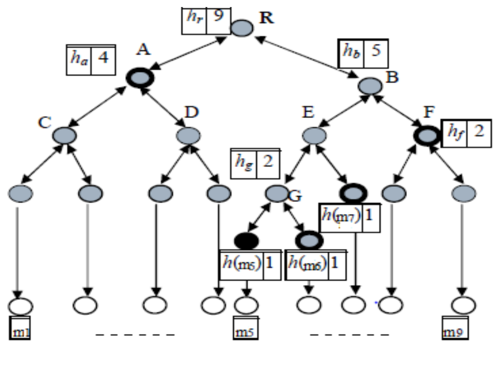

Fig.6: The DMHT with relative index

3)The Storage Security Model:

The Storage Security Model is used to monitor the security of the stored file. It is based upon the concept of Provable Data Possession model (PDP). PDP model is a simple challenge-response model. In this, the client challenges the server and the server provides proof in return for the respective challenge. The client checks the proof and ensures correctness of stored data. In our security system, there are two phases viz. setup phase and the integrity phase.

- **Setup Phase:**

In this phase, a file F= {m_1, m_2...m_n} is generated by the client, which is a finite collection on n blocks. Using the key generation algorithm, the keys are generated. Here we make use of a key generation algorithm to generate a unique key (secret key) for each user.Fig.7 depicts the overall flow of this process.

This phase consists of five steps. Firstly, using the secret key and the hash algorithm, the client generates the signature (tag) for each file block as Ti= $(H\ (m_i).gm_i)^{sk}$. Secondly, it generates a collection of signatures of file blocks ϕ= {T_i} called as a signature set. In the third step, a Dynamic Merkle Hash Tree (DMHT) is constructed for the file. In the fourth step, it signs the root R of the DMHT using the secret key as $sig_{sk}(H(R))$ =H(R). In the last step, the client advertises {F, ϕ, $sig_{sk}(H(R))$} to the server and deletes F and $sig_{sk}(H(R))$ from its local storage.

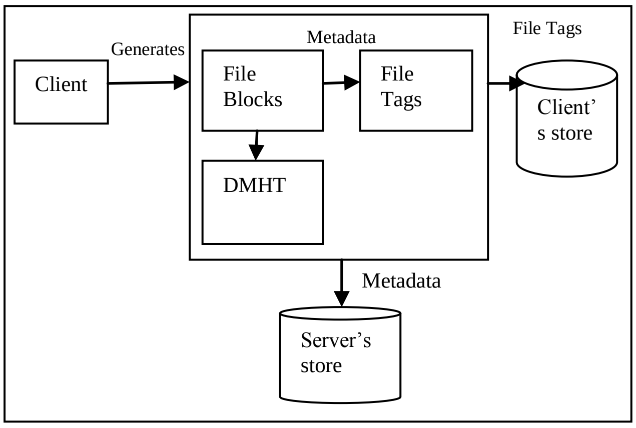

Fig.7: Pre-processing File Blocks

- **Integrity Phase:**

The integrity verification process is shown in Fig.8. After the challenge {i,a_i} is generated, the client sends it to the server and the server generates proof for the corresponding challenge. The TPA verifies the proof. Proof contains the signature of the root of the respective file, set of tags and the file name F. TPA compares all these details from the previously stored information. Any changes made to a file are reflected in the proof. If the proof matches to the metadata, then the file is considered to be in integrated state otherwise an alert message is sent to the user.

4) Implementation Details

- **Method for searching (i-th leaf node)**

First compare i with index (n) of the root node. If i is greater than n, then FALSE is emitted as output. Otherwise, let k=i and (h_a,n_a) be the left subtree and (h_b,n_b) be the right subtree of the current root. Now compare k with the relative index of left child, if $k \leq n_a$, then k lies in the left subtree and this algorithm is then used to find the node in left subtree, otherwise it is in the right subtree. If it lies in the right subtree, let $k=k-n_a$ and use this algorithm to find the node in the right subtree. Repeat this process until k=1 i.e. a leaf node is reached. During the process of searching i-th leaf node, the sibling of current node can be recorded for AAI (Ω_i) by the server.

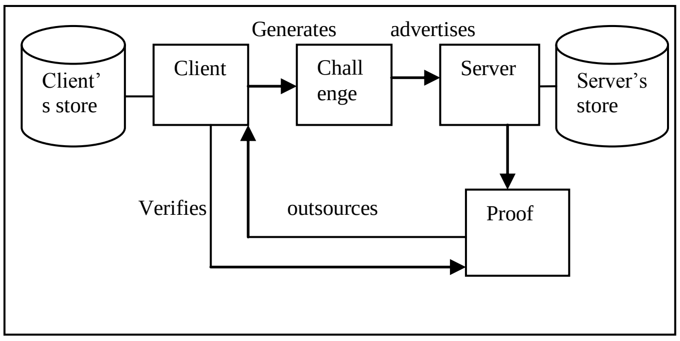

Fig.8: Integrity checking process flow [14]

- **Dynamic Data Operations:**

(1) Data Insertion:

Suppose the client wants to insert a data block say m* after a block mi –the i-th block. To do this, the client generates a signature for m* using the secret key sk and generator g as T = $(H(m^*).gm^*)^{sk}$. Then it constructs an update request as update= (I, i, m*, T*) and sends it to the server. When server receives this request, it executes the update operation. (i) The server stores the block m* and leaf node h(H(m*)). (ii) In DMHT, it finds $h(H(m_i))$, reserves Ωi and then inserts the leaf node h(H(m*)) after the i-th node. An internal node $h=h(h(H(m_i)||1) \ || \ h(H(m^*))||1)$ is added to the original tree and the index of this node is 2. Then, the information of all the nodes which lie in the path from this internal node to the root are modified i.e. their hashes are recalculated and the index is leaf node+1. (iii) Based on the updated DMHT, a new root R` is generated. After the update operation is successfully completed, the server sends a proof of this operation to the client. $P_{update} = (\Omega_i, H(m_i), sig_{sk}(H(R)), R`)$, where Ω_i is AAI for authentication of block mi in the old tree. When client receives this proof, it generates root R using $\{\Omega_i, H(mi)\}$ and then authenticates AAI or R. If the result for this authentication is TRUE, then the client is free to check if the server has performed the insertion by computing the value of new root using $\{\Omega_i, H(m_i), H(m^*)\}$ and then comparing it with R`. If the values of new root and R` does not match, the output is given as FALSE otherwise TRUE. If it is true, the client signs the new root by sk as $sig_{sk}(H(R`))$ and sends it to the server for updation.

The Fig.9 describes an example of insertion of a data block m* after the block m_5. The bold and underlined numbers in the figure indicates that the value was modified.

(2) Data Deletion:

Data deletion operation is just the opposite of data insertion operation and has similar process. An example is shown in fig. 10 based on fig. 6.

Fig.10 shows the resulting DMHT after the deletion of data $h(m_5)$ from the DMHT in Fig.6. Then, the information of all the nodes which lie in the path from the node to be deleted to the root are modified i.e. their hashes are recalculated and the index is leaf node-1.

(3) Data Modification:

The data modification operation does the work of replacing the data and so the structure of the tree does not change. The details of the protocol procedures are same as those of the data insertion.

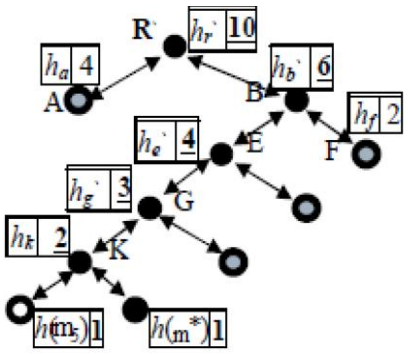

Fig.9: Example of block insertion operation in DMHT

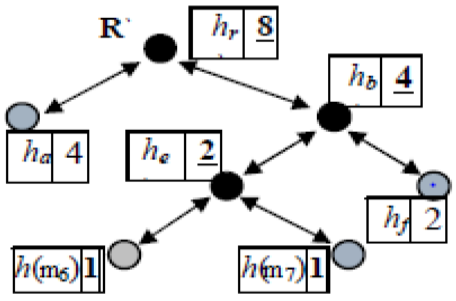

Fig.10: Example of Data Deletion operation in DMHT

4. CONCLUSION & FUTURE SCOPE:

Ensuring cloud data storage security is a necessary action to safeguard client's data and elevate the service quality. In this paper, we perform a survey on various ways used to ensure data integrity. Along with this, we propose a system based on AES using which DMHT is constructed. We hereby present a way of generating such a DMHT which provides integrity verification, data dynamics and also allow the server (prover) to search the specified data blocks efficiently.

This system can be enhanced in a lot of ways. A backup and recovery system can be added in order to recover the lost or corrupted files from the backup section. During recovery process, instead of fetching entire file from the backup data base, recovery can be done by fetching only the infected block. This will greatly reduce the communication cost. Secondly, a dynamic auditing method can be implemented so that the auditor can periodically check for the files without waiting for the request from the client. This method will completely remove the client's overhead. The client will simply get a notification if any of his files are lost or corrupted and asked for the recovery option. Also instead, the auditor can simply correct the content and maintain the client's data safely. Thirdly, the system can be designed to support multiple auditors so that if an auditor temporarily goes down, the other one can provide his service to the client without delay.

5. REFERENCES

[1] Y. Deswarte, J. Quisquater, and A. Saidane, "Remote integrity checking", In Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS'03), November \2003.

[2] T. Schwarz and E.L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely

administered storage”, In Proceedings of ICDCS ’06. IEEE Computer Society, 2006.

[3] G.Ateniese, “Provable Data Possession at Untrusted Stores”, Proc. 14th ACM Conf. Computer and Comm. Security (CCS’ 07), 2007.

[4] A. Juels, “Pors: Proofs of Retrievability for Large Files,” Proc. 14th ACM Conf. Computer and Comm. Security (CCS ’07), pp. 584-597, 2007.

[5] G.Ateniese, “Scalable and Efficient Provable Data Possession”, Proc. Fourth Int’l Conf. Security and Privacy in Comm. Networks (SecureComm ’08), 2008.

[6] C,Erway, A.Kuocu, C. Pamanthou, R.Tamassia, “Dynamic Provable Data Possession”, Proc. 16th ACM Conf. Computer and Comm. Security (CCS’09),2009.

[7] Cong Wang, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing”, IEEE Transactions on Parallel and Distributed Systems, May 2011.

[8] C.Wang, Q.Wang, Kui Ren, Wenjing Lou, “Ensuring Dynamic Data Storage Security in Cloud Computing”, Proc. 17th Int’1 Workshop Quality of Service (IWQos’09),2009.

[9] P. Golle, S. Jarecki, and I. Mironov “Cryptographic primitives enforcing communication and storage complexity”. In Financial Cryptography, pages 120-135, 2002.

[10] L. Chen and H. Chen,”Ensuring Dyanmic Data Integrity with Public Auditing for Cloud Storage”, In Proc. Of International Conference on Computer Science and Service System (ICSSS’ 2012), 2012.

[11] D.G.Feng, M. Zang, Y. Zang and Z. Xu,”Study on cloud computing security”, Journal of Software, vol.22 (1), pp. 71-83, 2011.

[12] L.M. Kunfam, “Data Security in the world of cloud computing”, IEEE Security and Privacy, vol.7 (4),pp.61-64,2009.

[13] B. Waters and H.Shacham, “Compact proofs of Retrievability”, Proc.14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT’ 08), pp.90-107, 2008.

[14] M. Venkatesh, “Improving Public Auditability, Data Possession in Data Storage Security for Cloud Computing”, ICRTIT-IEEE 2012

[15] Elminaam, Diaa Salama Abdul, Hatem Mohamed Abdul Kader, and Mohie Mohamed Hadhoud. "Performance Evaluation of Symmetric Encryption Algorithms." IJCSNS International Journal of Computer Science and Network Security 8.12 (2008): 280-286.

[16] Simar Preet Singh, and Raman Maini, “COMPARISON OF DATA ENCRYPTION ALGORITHMS”, International Journal of Computer Science and Communication (IJCSC), Vol. 2, No. 1, January-June 2011, pp. 125-127