International Journal of Engineering Research
Volume No.5, Issue No.7, pp : 622-626
ISSN:2319-6890)(online),2347-5013(print)
1 July 2016

Secure Fine grains Updates in Cloud using Public Auditing

Tanuja.Sali , Mansi Bhonsle
G.H.R.C.E.M Wagholi, Pune , Savitribai PhulePune University, Pune, India
tanujasali@rediffmail.com, mansi.bhonsle@gmail.com

Abstract: Data security in computing of computers on internet has become the most major worries as user of cloud cannot control the data directly. Data integrity is to be verified without actual possession of data in present system. This paper proposes a scheme for authorized public auditing and fine grained updates which reduces communication overhead for small update request. Results of experiments conducted till now shows that our scheme can offer improved security, flexibility, and less overhead for processing big data applications.

Keywords- Authorized public auditing, Fine grains data, Dynamic data updates, Cloud computing

I. Introduction

CLOUD computing is considered as the most innovative information technology in the some recent years. Cloud computing offers three types of computing services which are IaaS, PaaS and SaaS. Some of examples of cloud are IBM intelligent Cloud, Amazon AWS, and Microsoft Azure. Cloud delivers computing services and resources in a pay as you go mode, to users on a range from individual to firms, organizations and enterprises all over the world.

Data security and integrity is the main concern in cloud computing where the user cannot directly control the data placed in the cloud. However in this paper, there is the major problem of proving and verifying the data, for its consistency and accuracy and also for storing big data in cloud. This method of proving data integrity is called auditing of data and this process of auditing is done by third person who is trusted one. This TPA is responsible for generating random challenges, proof verification and maintains logs. This auditing should be done on regular basis for those users whose data demands require high level of security.

This existing auditing scheme are inefficient and have potential risk such as security threats and misuses in unauthorized auditing requests and also it is inefficient in processing small updates of data. In this paper we describe potential security problem which is supported by public auditing, and is one of the most secure and robust schemes. Here surplus authorization is done between client/user, TPA and cloud storage server (CSS). Our system also supports small request for changing updates, which will indeed benefit the cloud storage server by increasing its scalability and efficiency. To obtain this, our scheme uses a data segmentation method which is flexible for Partitioning of data.

II. Material and Methodology

Scalability, reliability and elasticity are some of the key advantages of cloud as compared to the traditional system. Security and protecting privacy of dynamic data is considered as a fundamental part in cloud computing. This paper focuses on repeatedly happening small changes of data, as these updates occurs in most applications of cloud such as social online network and also in business transactions. Here clients may need to break big dataset blocks into data blocks which are small in size and store them on different servers for preserving reliability, privacy and for efficient processing purposes. Data privacy and security is the major problem of cloud.

A. Existing System

Public data verification is generally done by PDP and POR schemes. Client, CSS and auditor are the three participants involved as shown in figure. The given system is the traditional PDP system. The CSS has to just store the data/files directly without maintaining any security or keys. TPA gets blocks of data instead of keys. TPA gets full access of data and hence the data is not secured. The TPA can steal or sold the data. Thus both TPA and CSS are semi trusted to cloud.
In this model, the challenge message for verifying proof for a set of file blocks can be sended by anyone to CSS, which may sometimes lead to fraud exploits in practice. Thus the security requirements cannot be fulfilled by traditional PDP. In existing system some of public schemes for auditing can support full updates for dynamic data, where deleting data, inserting data, and modification of data are done on one or more same size blocks which is known as coarse-grained update. Here updates are done only on 16K data block size. Thus this system is not supporting data blocks of different or variable sizes. Due to this reason sometimes partially allocated storage is wasted and it cannot be reused until the whole block is deleted.

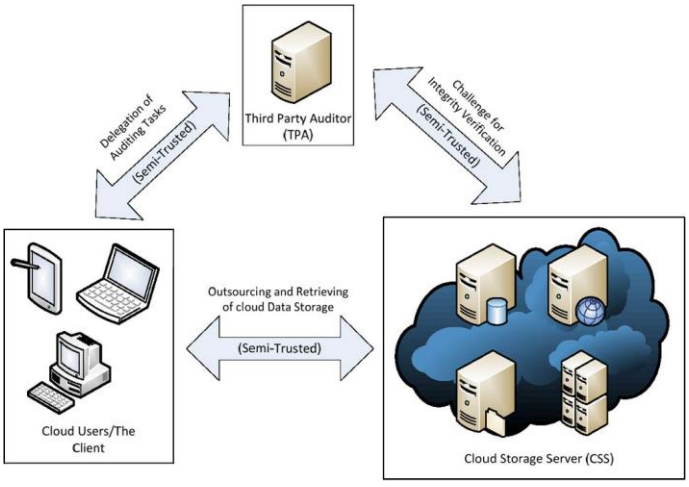

Fig. 1. Existing system diagram

ISSN:2319-6890)(online),2347-5013(print)
1 July 2016

B. Proposed System

The Proposed system analyzes various types of update requests on dynamic data of varied size of files. Additional authorized authorization process is incorporated in our scheme to eliminate malfunctioning of unauthorized challenges from unknown users during auditing process. This scheme also investigates how the efficiency can be improved while updating and verifying frequent minor updates which prevails in many cloud applications.

As shown in the Fig. 2 below we have

Users: Selects files to upload on the CSS or download the files from CSS and it also do modifications in files and data. The user here is a thin user having no burden of processing of data or files.

Central Server: CS receives the files and partitions the files. Then it extracts digital signature of each files, and generates confidential hash key for each block of partition which are given to TPA. Finally it encrypts the data using secret confidential keys and then stores the partition sequence, keys for signature and file attributes on its own storage server. After that it stores actual encrypted data blocks in the respective storage server of the cloud. It maintains different storage server to store the data. Thus if anyone of the CS or Storage server is hacked, the and malicious user will get only keys but not data or he will get only continuous slots of data with no keys to recognize the data blocks. Hence it is fully safe and can be trusted fully.

Storage Server: The storage server receives the file partitions and then stores it.

TPA: TPA stores hash keys and maintains logs. It generates random challenges for the CS; the CS will demand the file from respective cloud and then send the hash keys to TPA. Then TPA compares hash keys with the stored ones. If any data or file is corrupted message will be sent to corresponding user.

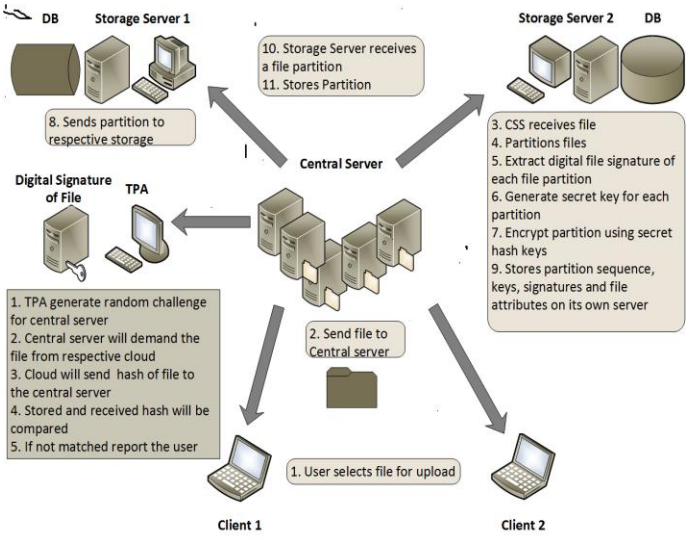

Fig 2. Proposed system diagram

.

Fine grains updates

Our scheme supports changing modifications for the variable sized data. It also supports five types of operations on different blocks of data:

- Partial Modification (PM):In partial modification, a linear successive part of a certain block will be updated
- Whole block modification (M):Here entire data block will be modified by a new set of data block
- Block deletion (D) :In block deletion, full data block will be erased from the structure of tree
- Block insertion (J): Here a full or entire block has to be created on the tree structure to hold values of data which is newly inserted
- Block Splitting (SP) :In this, some data in the block is removed out to form a new space for inserting the new data .

Our proposed system has less communication overheads than the existing system. Our scheme has three steps which are described as follows:

1. Setup:

The client produces keys via KeyGen for generating keys and FileProc for performing modifications and segmentation, and then it will send the data to CS. The CS will divide the files into small slots of partitions by using RHMT, and then encryption of data will be done by using AES algorithm. Client will approve the TPA by giving a value sigAUTH using SHA algorithm.

Key generation: The client creates a confidential hash value $\alpha \varepsilon Zp$ and a generator g of G, to compute the value of $\upsilon = g^{\alpha}$. A secret key pair (spk; ssk) is selected from a assured safe signature process whose algorithm for signing is denoted as Sig( ). This algorithm will give output as the secret key (ssk, α) and public key pk as (spk; υ; g).

FileProc (F; sk; SegReq): Here data fragmentation technique is done. Where the whole data Smax is divided into some no of data blocks whose size is equal except the last one which sometimes may be unequal depending on the even or odd value of Smax[2]. Here output (F; ϕ; T; R; sig; t) is obtained where F is file tagF=mi, j where I ε [1,l]; j ε[1,s];and si ε[1,smax], i, ϕ is ordered set ϕ= α , i ε [1,l] , sig= (H(R))α , t=(name ‖ n ‖ u ‖Sigssk(name ‖ n ‖u):

2. Verification for Data Updating:

This process is done between client and server. Here we will consider five different types of operations as PM, M, D, J and SP. The CSS performs update request for fine-grains data of the client via PerfUpdate, then VeriUpdate is run by client to examine whether CSS has performed the updates correctly on data blocks.

- Pupdate = (fmi; i;R; sig)and the updated file F

International Journal of Engineering Research
Volume No.5, Issue No.7, pp : 622-626
ISSN:2319-6890)(online),2347-5013(print)
1 July 2016

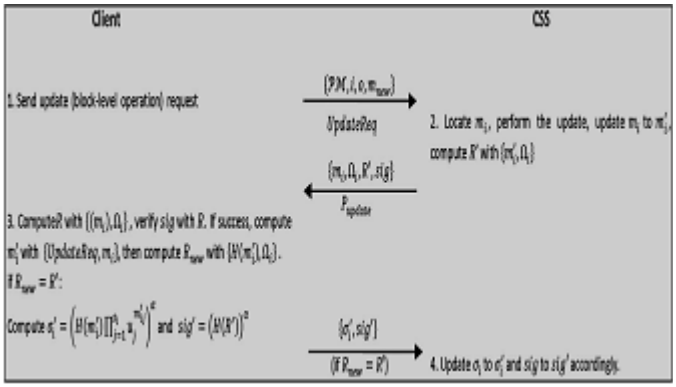

Fig. 3. Verifiable updates of data diagram

- Then CSS will forward Pupdate to the client.
- PerfUpdate (UpdateReq; F): Here in this step CSS raises request for update and gets (PM; I;o; mnew). When Type of request is PM, CSS will change both mi and Ti values, and then it outputs Pupdate = (mi;Wi;R0; sig) and the updated file F0 is sended to the client by CSS.
- After receiving Pupdate request client performs request for VeriUpdate (pk, Pupdate) where the client evaluates mi using (mi; UpdateReq), then parses Pupdate to mi; i; R;sig to compute R (Hr).It verifies sig using Hr, and checks if Rnew = R. If any of these two verification matches fails, then the output will be FALSE and it will return to CSS, otherwise the output will be TRUE. Then the client computes _ and sig and then sends these_, sig to the CSS. The CSS then changes _i to _i and sig to sig and also deletes F

3. Challenge generation, Proof generation and Verification of the Proof:

Here it describes how the verification or testing of integrity is done by TPA to the data stored on CSS. TPA conducts three functions which are ChallengeGen, ProofGen and VerifyGen.

1. Challenge generation:
ChallengeGenAcc; pk; sigAUTH: According to the accuracy and consistency needed, TPA will decide to verify some c blocks out of the total blocks (l). Then, TPA gives a challenge message chall = sigAUTH; VID) PKCSS;(i; vi) where VID
is TPAs ID. TPA then forward chall to CSS.

2. ProofGen (pk; F; sigAUTH; ϕ; chall):
Here CSS runs the ProofGen after receiving the chal.CSS verifies sigAUTH with auditors VID and the clients public key spk, and the resultant output is REJECTED if it fails. If the condition satisfies then, CSS will compute k =Σi εI (vi, mik); k ε(1;w) and σ=σi for composing of the proof P=kkz ε[1,w],H(mi),Ωi,i ε I,sig. Then this proof will be sent by CSS to TPA.

3. TPA will execute the VerifyGen request after receiving the P from CSS. VerifyGen(pk; chal; P): TPA will compute R using H(mi);i and then verification will be done using sig using public keys g and by comparing e(sig;g) with (HR;υ). If they match then the verification is done successfully.

Ranked Hash Merkle tree for data Fragmentation
Partitioning of files is done by extended Hash Merkle Tree (HMT) which is known as Ranked Hash Merkle Tree (RHMT). Figure shows ranked hash merkle tree

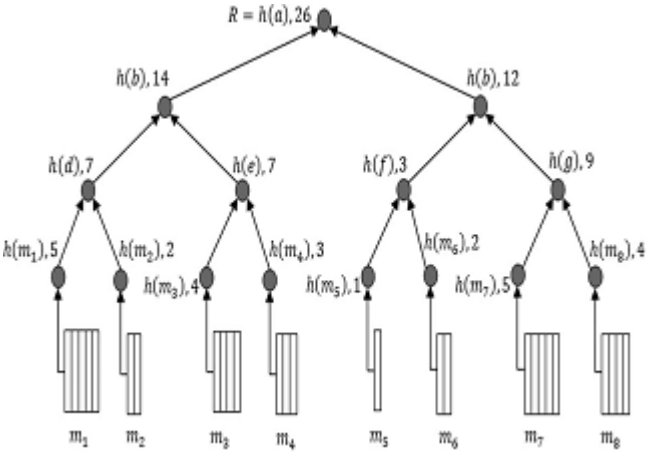

Fig. 4. Ranked Hash Merkle Tree example

1. Here each node of RMHT will have at the max two leaf nodes or child nodes which resembles to the binary tree.
2. In RMHT, information present in any single node N is written as H; rN where H is a confidential secret value and rN is the rank of this node.
3. Tree is generated in following way:
For every leaf node LN having message mi, we have H= {h (mi), r (LN)=si } The parent nodes N1 and N2 are calculated as N1 = {H1; rN1} and N2 = {H2; rN2},and Np is constructed as Np = {h(H1 ‖ H2); (rN1 + rN2)} where k is a or operator or also called as concatenation
operator. A leaf or child node message will now contain mi's=AAI where i is a set of secret values chosen by every node from its parent node in order to compute the root value R
through mi; i

Algorithm for encryption of files:
Our scheme use AES (Advanced encryption system) for encryption. It has different keys of length bits and it encrypts data of 128 bits in 10 rounds, data of 192 bits in 12rounds and data of 256 bits in 14 rounds. AES encryption is fast and flexible and size of output ciphertext file is same as the size of original data file. [12]
Ciphertxt (byte[] input1, byte[] output1) f
Byte [4,4] State1;
copy input1[] into State1[] AddRoundKey
for (round = 1; round < Nr-1; ++round) f
SubBytes ShiftRows MixColumns AddRound-
Key g
SubBytes ShiftRows AddRoundKey
copy State1[ ] to output1 [ ] g

Algorithm for generating Hash key:
Our proposed scheme uses SHA/BLS algorithm to generate Hash keys. An algorithm that takes as input the strings of any length and reduces it to a unique constant length string is called

ISSN:2319-6890)(online),2347-5013(print)
1 July 2016

Hash function. It is used to protect the integrity of data and message, and is also used to check the password validity.

Steps for Hash are:

1. Initialize some (say five) variables say h0, h1,h2, h3, h4
2. Select any string for generating secret codes
3. Break the string into different characters.
4. Convert all the characters to ASCII codes.
5. Convert all the numbers into binary digits.
6. Add zero at the end.
7. Then append one at the end
8. Again append original message at the end.
9. Chunk the message
10. Break the chunk into different words
11. Extend up to 80 words where every step at the end will be repeated until a certain condition is valid.
12. All the given words must be XORed simultaneously.
13. Perform a left bit rotation by a factor of one unit.
14. Thus the resultant output is obtained by using function 'f' equal to: (B AND C) or (!B AND D).

Mathematical model

Set Theory

Let X = {G, I ,T,O,F}

Where,

- G is a global set containing users, servers, File, partitions, keys and challenges.
- I is set of inputs/challenges T is updates/ modifications/ processing done on I.
- is the outcome obtained after processing.
- F is the final state after the processing.

Let G be considered as a global set

G = {U, S, F, P, K, C}

U is set of users where u = {u1, u2, u3... u n} where n=infinity

S is set of servers where S = {s1, s2, s3... sn}

F is set of files where f = {f1, f2, f3..., fn}

P is set of partitions where P = {p1, p2, p3...pn}

K is set of keys where K = {k1, k2, k3... kn}

C is a set of random challenges C = {c1, c2, c3... cn}

Homomorphism

SDB' = Register User (user details);

Yes/no ⟵ login (user id, password);

Up (Tpa) ⟵ up load files (file details);

{p1, p2... pk} ⟵ partition files (files);

{Ds1, Ds2... Dsk} ⟵ Calculate SHA {P1, P2... Pk}

{Ep1,Ep2,...,Epk} ⟵ encrypt{P1,P2,...,Pk} {K1,K2,...,Kn}

Storage ⟵ partition positions< >;

{Dp1,Dp2,..., Dp} ⟵ Decrypt {Ep1, Ep2...Epk} {K1, K2,Kn}

III. Results and Tables

Experimental results reveal that our scheme has less storage burden for small data request of insertions and updates. Our proposed schemes support changing updates on small various slots of data, without which every small insertion would have caused creation of a whole new block and update of related MHT nodes; hence our scheme is efficient.

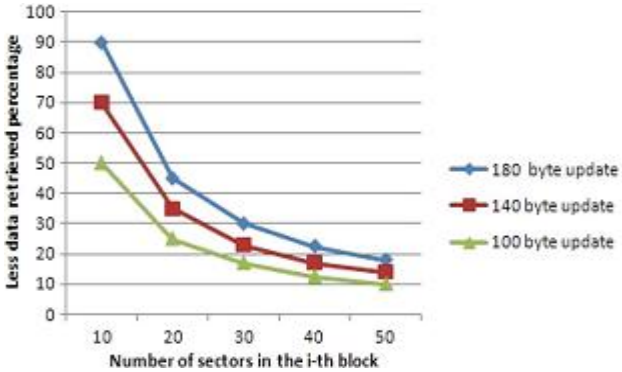

Fig 5. Percentage in saving of communication overhead

Thus we can see that our scheme incurs significantly lower storage overhead and also dramatically reduces communication burdens as compared to the existing scheme. However the parameter smax should be set according to different efficiency demands and different data sizes in storage or communications. The number of Sector is one of the most influential metrics to overall performance.

IV. Conclusion

Our scheme has done through analysis for data updates on by considering all the different types of operations on fine-grained data and hence putted forth a system that supports authorized public auditing and update requests for fine grain data. Our scheme drastically reduces communication work for verifications of small updates. Theoretical and experimental result demonstrated that this system offers flexibility, enhanced security and is less burdening for applications of big data with a number of frequently occurring small dynamic updates. Due to support for variable block modification of data the storage consumption problem is reduced. Also our scheme offers an increased level of security due to authorized integrity checking methods

Acknowledgement

I express my sincere thanks to M.E. Coordinator Prof. Mansi Bhonsle for support for successful completion of this seminar would not have been possible. I am also thankful to Prof. Dr. Shah (Principal, GHRCEM) for support to complete my project. I thank my classmates and my family for supporting.

References

i. Chang Liu, Jinjun Chen,Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates, IEEE Transaction on Parallel and Distributed System, vol. 25, no 9,September 2014.

ii. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic Cloud Computing and Emerging IT Platforms: Vision, Hype, Reality for Delivering Computing as the 5th Utility, Future Gen.Comput. Syst. Vol 25, no. 6, pp. 599-616, June 200.

iii. Kan Yang,An Efficient and Secure Dynamic Auditing Protocol for Data storage in Cloud Computing, IEEE Transactions on Parallel And Distributed Systems, Vol. 24, NO. 9, September 2013.

ISSN:2319-6890)(online),2347-5013(print)
1 July 2016

iv. M.Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.Katz, A.Konwinski,G. Lee, D. Patterson, A. Rabkin, I. Stoica, andM. Zaharia, AViewof Cloud Computing, Commun. ACM, vol. 53, no. 4, pp. 50-58,Apr. 2010.

v. Jian Liu, Kun Huang, Hong Rong, HuimeiWang, and Ming Xian, Privacy-Preserving Public Auditing for Regenerating- Code-Based Cloud Storage,IEEE Transactions on Information Forensic and Security, vol. 10, no. 7, July 2015

vi. Boyang Wang, Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud, IEEE Transactions on Cloud Computing vol. 2, no. 1, March 2014

vii. Jiawei Yuan and Shucheng Yu,Public Integrity Auditing for Dynamic Data Sharing with Multiuser Modification, IEEE Transactions on Information Forensics and Security, vol. 10, no. 8, August 2015.

viii. Hasan and Burkhard Stiller, SLO Auditing Task Analyse Decomposition,and Specification, IEEE Transactions on Network and Service Management, vol. 8, no. 1, March 2011.

ix. G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, Remote Data Checking Using Provable Data Possession, ACM Trans. Inf. Syst. Security, vol. 14, no. 1, May 2011, Article 12.

x. Chang Liu, Rajiv Ranjan, Chi Yang, Xuyun Zhang, Lizhe Wang, Top Down Levelled Multi-Replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud, IEEE Transactions on Computers, vol. 64, no. 9,September 2015.

xi. C. Erway, A. Ku pcu , C. Papamanthou, and R. Tamassia, Dynamic Provable Data Possession, in Proc. 16th ACM Conf on Computation. and Communication. Security (CCS), 2009, pp. 213-222.

xii. Wikipedia on Web of Things