Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Multi Client Support Third Party Auditor (TPA) for Cloud Data Integrity and Security

Mrs. Vrushali R. Desale , Dr. Pradeep K. Deshmukh
Rajarshi shahu College of Engineering, Pune
Computer Enng. Dept., pune University, India

Abstract— Cloud computing provides many benefits for the users to create and store data in the remote servers thereby utilizing fewer resources in client system. In this paper we have presented the extended approach for cloud computing data security and integrity verification with data dynamics under the multiple users. Previously the efficient approach is presented with single client which is gets connected with TPA and CSS (Cloud Storage Server) for data dynamics and integrity verification. However under the cloud environment there may be multiple clients connected with CSS; hence we are aiming to utilize the approach efficiently for such cases by using the concept of synchronization of multiple clients with TPA and CSS. This paper is focusing over security and integrity of data stored at untrusted or semi-trusted cloud storage server under multiple clients. Here importantly it needs to monitor the tasks related to data auditing efficiently as per the clients requests. The proposed system is capable of supporting both public auditability as well as data dynamics. Merkle Hash Tree is also used for improving the level of authentication. The new security model for data blocks security is presented.

Keywords— Data Security, cloud computing, Third party auditor, cloud storage server, data dynamics.

I. INTRODUCTION

Currently there are many trends are gap up the time of Cloud Computing, that is an Internet-based development and use of technology. The ever cheaper and additional powerful processors, alongside the "software as a service" (SaaS) computing design, are remodelling data centers into pools of computing service on an enormous scale. Meanwhile, the increasing network information measure and reliable nonetheless versatile network connections build it even potential that clients will currently subscribe top quality services from knowledge and software system that reside only on remote knowledge centers. Though visualized as a promising service platform for the net, the new knowledge storage paradigm in "Cloud" brings concerning several difficult style problems that have profound influence on the protection and performance of the general system. One in all the most important issues with cloud knowledge storage is that of knowledge integrity verification at untrusted servers. what's additional serious is that for saving cash and space for storing the service supplier may neglect to stay or deliberately delete seldom accessed knowledge files that belong to a standard shopper. Think about the massive size of the outsourced electronic knowledge and also the client's strained resource capability, the core of the matter is generalized as however will the client realize associate degree efficient thanks to perform periodical integrity verification's while not having the native copy of knowledge files. Considering the role of the booster within the model, all the schemes given before constitute 2 categories: non-public auditability and public auditability. Though themes with non-public auditability can do higher scheme potency, public auditability permits anyone, not simply the shopper (data owner), to challenge the cloud server for correctness data storage whereas keeping no non-public information. Then, end users are able to delegate the analysis of the service performance to associate degree freelance third party auditor (TPA), while not devotion of their computation resources. Within the cloud, the clients themselves are unreliable or might not be able to afford the overhead of playing frequent integrity checks.

To overcome the issues associated with cloud security and database integrity, recently the TPA based approach is presented which is having following features:

(a) To encourage the general public auditing system of knowledge storage security in Cloud Computing, and propose a protocol supporting for absolutely dynamic knowledge operations, particularly to support block insertion, that is missing in most existing schemes;

(b) To increase the theme to support ascendable and economical public auditing in Cloud Computing. Particularly, the theme achieves batch auditing wherever multiple delegated auditing tasks from totally different user are performed at the same time by the TPA.

(c) To prove the protection of the projected construction and justify the performance of the theme through concrete implementation and comparisons with the progressive.

However, this method is only evaluated and performing well under the single client connected with TPA and CSS. Hence we are presenting the extended algorithms of above methods by synchronous connections and monitoring of

multiple clients for their database integrity verification and security at cloud server.

II. RELATED WORK

Recently, much of growing interest has been pursued in the context of remotely stored data verification. Ateniese et al. [1] are the first to consider public auditability in their defined "provable data possession" (PDP) model for ensuring possession of files on untrusted storages. In their scheme, utilize RSA based homomorphic tags for auditing outsourced data, thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems. In their subsequent work [2], Ateniese et al. propose a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported. In, Wang et al. consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [2], they only consider partial support for dynamic data operation. Juels et al. [10] describe a "proof of retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on archive service systems. Specifically, some special blocks called "sentinels" are randomly embedded into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [2], the number of queries a client can perform is also a fixed priori, and the introduction of precomputed "sentinels" prevents the development of realizing dynamic data updates.

In addition, public auditability is not supported in their scheme. Shacham et al. design an improved PoR scheme with full proofs of security in the security model defined in [10].

They use publicly verifiable homomorphic authenticators built from BLS signatures [4], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static data files. Erway et al. [9] was the first to explore constructions for dynamic provable data possession. They extend the PDP model in [1] to support provable updates to stored data files using rank-based authenticated skip lists. The scheme is essentially a fully dynamic version of the PDP solution. To support updates, especially for block insertion, they eliminate the index information in the "tag" computation in Ateniese's PDP model [1] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear. Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing. Two basic solutions (i.e., the MAC-based and signature based schemes) for realizing data auditability and discuss their demerits in supporting public auditability and data dynamics.[3,11] Secondly, generalize the support of data dynamics to both proof of retrievability (PoR) and provable data possession (PDP) [5,10] models and discuss the impact of dynamic data operations on the overall system efficiency both. In particular, emphasize that while dynamic data updates can be performed efficiently in PDP models more efficient protocols need to be designed for the update of the encoded files in PoR models.

III. PROGRAMMERS DESIGN

A. *Contribution*

In this paper we present a framework and an efficient construction for seamless integration of these two components in our protocol design. Our contribution can be summarized as follows:

(a) We propose a general formal model with public verifiability for cloud data storage, in which block less verification is achieved.

(b) We equip the proposed construction with the function of supporting fully dynamic data operations, especially to support block insertion;

(c) We prove the security of our proposed construction and justify the performance of our scheme through concrete implementation and comparisons with the state-of-the-art.

(d) We improve the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication to achieve efficient data dynamics.

(e) We further explore the technique of bilinear aggregate signature to extend our main result into a multiuser setting, where TPA can perform multiple auditing tasks simultaneously.

(f) Extensive security and performance analysis shows that the proposed scheme is highly efficient and provably secure.

B. *Design Goals*

Our design goals can be summarized as the following:

(a) Public verification for storage correctness assurance: to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand;

(b) Dynamic data operation support: to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public verifiability and dynamic data operation support;

(c) Block less verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for both efficiency and security concerns.

(d) Stateless verification: to eliminate the need for state information maintenance at the verifier side between audits throughout the long term of data storage.

(e) Multi-User Support by TPA.

C. ***Mathematical Model***

As in this approach we enhance the theme with express and efficient multi user based dynamic knowledge operations for data storage security in Cloud Computing. Therefore, it's crucial to contemplate the dynamic case, wherever a user might need to perform numerous block-level operations of update, delete and append to switch the information file whereas maintaining the storage correctness assurance. The simple and trivial way to support these operations is for user to transfer all the information from the cloud servers and re-compute the total parity blocks in addition as verification tokens.

1) Update Operation

In cloud data storage, typically the user may have to switch some knowledge block(s) stored within the cloud, from its current price fij to a replacement one, fij + Δ fij. We have a tendency to refer this operation as data update.

2) Delete Operation

Sometimes, when being hold on within the cloud, data blocks may have to be deleted. The delete operation, area unit considering may be a general one, during which user replaces block with zero or some special reserved data image. The delete operation is truly a special case of the information update operation, wherever the initial knowledge blocks is replaced with zeros or some planned special blocks.

3) Append Operation

The user might want to extend the dimensions of his hold on knowledge by adding blocks at the tip of the information file, we refer this as data append. We have a tendency to anticipate that the foremost frequent append operation in cloud data storage is bulk append, during which the user must transfer an oversized variety of blocks (not one block) at just the once. Dynamic operations area unit performed by constructing the matrix, wherever 0's indicate the blocks we want to vary and 1's indicate the unchanged blocks [8]. We produce cloud surroundings where user, TPA and cloud server are connected to one another. Publicly auditing system, the correctness of the information is checked by key generation, signature generation, generate proof and verify proof algorithms. Similarity appraiser with random masking is employed to realize privacy preserving auditing theme. The technique of bi-linear combination signature is employed to realize batch auditing. In cloud, the information doesn't stay static. We enhance the system which expresses dynamic operations in data blocks under the multiple clients.

4) Data Security Model

Following mathematical equations presents the data security in cloud computing:

D_f=C(NameNode); (1)

K_f=f * D_f; (2)

C(.): the visit of nodes;

D_f: the distributed matrix of file f;

K_f: the stste of data distribution in datanodes;

F: file, file f can be described as;

F={F(1), F(2), ……F(n)}, means f is the set of n file blocks.

F(i) ∧ F(j)=Φ, i!= j; j€ 1, 2,3,….n;

D_f is a zero-one matrix, it is L*L, L is the number of datanode.

Thus in order to provide the data security in cloud computing we are presented the below approaches:

$D_f^{'} = C_A$ (NameNode); (3)

$D_f = M.D_f^{'}$; (4)

$K_f = E(f) * D_f$; (5)

C_A(.) : authentic visit to namenode;

D_f : private protect model of file the distributed matrix;

M : resolve private matrix;

E(f) : encrypted file f block by block, get the encrypted file vector;

Following vein diagram (Fig. 1) is showing the same.

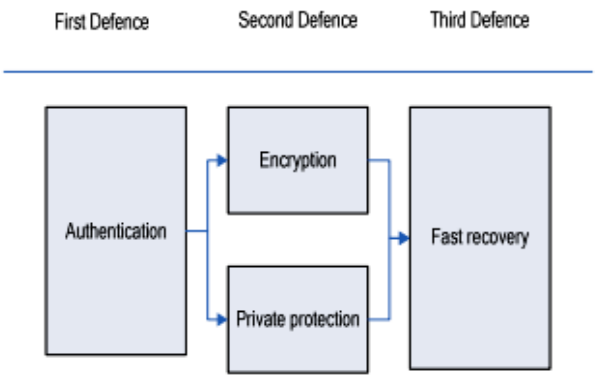

Fig. 1 Data Security Model

D. ***Algorithms***

- **KeyGeneration:** Run by client
  Input: None
  Output: public key rpk, secrete key rsk, generator g.

- **SignatureGeneration:** Run by client
  Input: File Blocks F, secret key rsk, generator g.
  Output: set of signature Φ.

- **GenerateProof:** Run by cloud storage server
  Input: Subset of file blocks mi, coefficient i
  Output: Proof P

- **VerifyProof:** Run by TPA
  Input: Proof P
  Output: Boolean value {TRUE, FALSE}

- **ExecUpdate:** Run by the server
  Input: file F, set of signature Φ, update query
  Output: new file F, new set of signature Φ, update proof.

- **VerifyUpdate:** Run by the client.
  Input : public key, update query, update proof
  Output: Boolean value TRUE, FALSE, and signature H(R')

1) Data Integrity Verification Algorithm for Multi-Client Environment.

- Start
- Multiple Clients synchronizing connection with Third party auditor as well as Cloud Storage Server.
- Client gets connected with Third party auditor for Request processing.
- Client generates a tag for each file block using signature generation algorithm
- A Merkle Hash Tree is constructed for each file block.
- The root R of the Merkle Hash Tree is signed using the secret key
- Client advertise file, set of signatures and computed root value to the server and deletes it from its local storage
- TPA generates a challenge and sends to the server
- Server generates a proof based on challenge, the proof contains auxiliary authenticate information using GenerateProof algorithm
- The server sends the generated Proof P to the client.
- The Third party auditor validates the proof by generating the root R, using verifyProof algorithm.
- After verification, the Third party auditor can determine whether the integrity is reached.
- Stop

2) Algorithm for Updating and Deleting Data Present in CSS for Multi-Client Environment

- Start
- Cloud Storage Server and Third party auditor Synchronizing Clients and Gets Connected according to their priorities.

- Client generates new Hash for tree then sends it to Cloud Storage Server
- Cloud Storage Server updates F and computes new root R'. Runs ExecUpdate algorithm
- Cloud Storage Server sends old root and new root to client.
- Client first verifies old root value to check whether CSS is updating the same block or not. Runs VerifyUpdate algorithm.
- Client computes new R and verifies the update block .If it fails outputs FALSE.
- Stop

E. ***System Design***

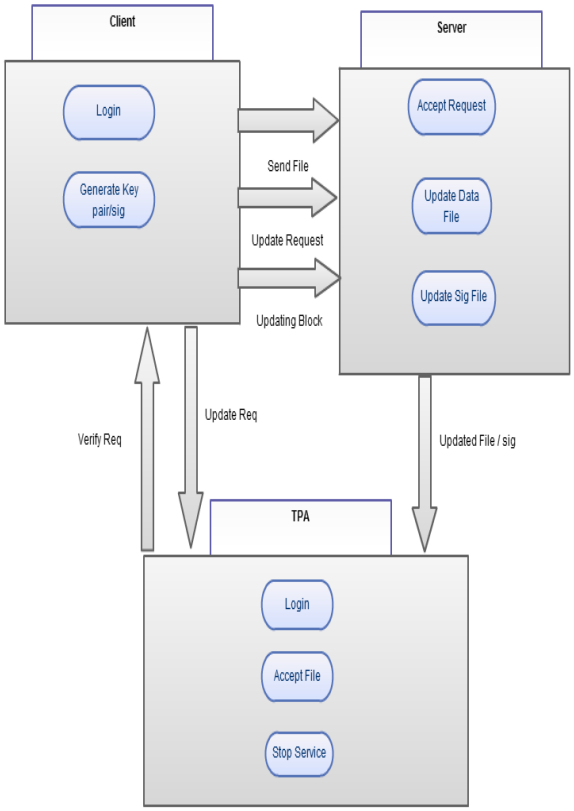

Fig. 2 System Flow Diagram

IV. RESULT AND DISCUSSION

Our experiment is conducted using Java on a system with an Intel dual core processor running at 2.4 GHz, 1 GB RAM. During the implementation we have made three parts which are major components of proposed approach such as clients, third party auditor and cloud server in order to store client's data. A sample set of 100 file blocks is processed and stored at server. The integrity verification is carried out using different set of challenges and verifies the proof generated by the server. The storage space for key generation, Merkle tree construction and signature generation take place in the constant order. In the integrity verification stage, the proof generation and verification take place in the O (log n) as the file block (mi) is used instead of index. Hence file index recomputation is avoided at each proof generation and verification process. Fig. 3 gives the server computation time using RSASS, compared with our algorithm under constant number of integrity challenges. Our algorithm is showing better performance than RSASS. In fig. 4, number of auditing tasks and auditing time per task are plotted in x and y axis respectively. Graph shows the comparison between individual auditing and batch auditing for 100 clients.

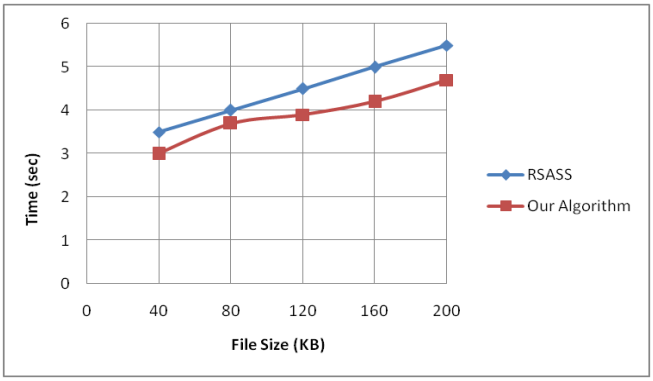

Fig.3 Server computation time

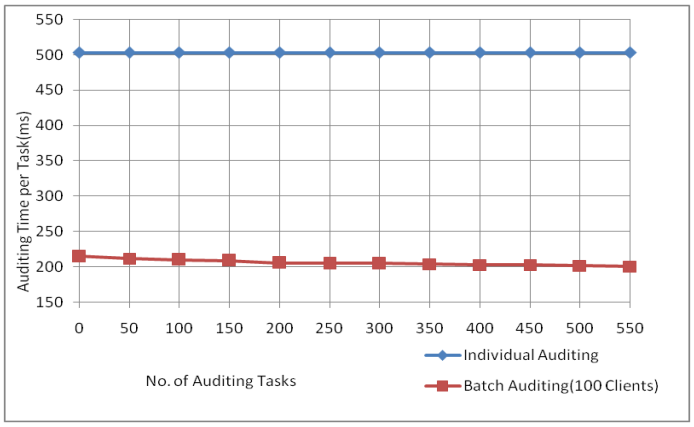

Fig.4 Auditing time per task for individual auditing and batch auditing

V. CONCLUSION

As in this paper we presented the extended approach for data dynamics and security of cloud database by proposing the model based on multiple clients. We identified the problem of existing system as it was working best in case of single client, however further it has to be extended in case of multiple clients. In order to get information dynamics that are effective, the prevailing proof of storage models is increased through manipulation of the development of classic Merkle Hash Tree for authentication of block tag. For supporting smart handling of multiple numbers of auditing tasks, the tactic of additive combination signature is more explored for extending the most result into a multiuser setting, wherever TPA during a position to perform multiple auditing tasks in a concurrent manner. This system is applicable to large public databases such as digital libraries, astronomy, medical archives etc.

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp. 598–609.

[2] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm'08, 2008.

[3] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou and Jin Li,"Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 5, MAY 2011

[4] A. L. Ferrara, M. Greeny, S. Hohenberger, M. Pedersen (2009), "Practical short signature batch verification", in Proceedings of CT-RSA, volume 5473 of LNCS. Springer-Verlag, pp. 309–324.

[5] H. Shacham, B. Waters (Dec 2008), "Compact proofs of retrievability", in Proc. of Asiacrypt 2008, vol. 5350, pp. 90–107

[6] M. A. Shah, R. Swaminathan, M. Baker (2008), "Privacy preserving audit and extraction of digital contents", Cryptology ePrint Archive.

[7] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Report 2008/186,Cryptology ePrint Archive, 2008.

[8] A. Oprea, M.K. Reiter, and K. Yang, "Space-Efficient Block Storage Integrity," Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS '05), 2005.

[9] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession" Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), 2009.

[10] . A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp. 584–597.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), 2009.

