 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Ensuring Effective Third Party Auditing (TPA) for Data Storage Security in Cloud Computing

[1]Moshe Dayan. Sirapangi, [2]G. B. V. Padmanadh
[1,2]Dept. of Computer Science, Kakinada Institute of Engg. and Tech (KIET), Korangi, Kakinada, AP, India

Abstract
Cloud computing is an internet based computing which enables sharing of services. Many users place their data in the cloud, so correctness of data and security is a prime concern. This work studies the problem of ensuring the integrity and security of data storage in Cloud Computing. Security in cloud is achieved by signing the data block before sending to the cloud. Signing is performed using BLS algorithm which is more secure compared to other algorithms.
This paper, proposes an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users data in the cloud. We rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction drastically reduces the communication and storage overhead as compared to the traditional replication based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the storage correctness insurance as well as data error localization, whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s). The new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete, append and insert.

Keywords
Data Storage, Public Auditability, Data Dynamics, Cloud Computing

I. Introduction

Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale.
Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) [1] are both well known examples. While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 is such an example [2].
From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons. Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature.
Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works [3–7]. These techniques, while can be useful to ensure the storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations. As an complementary approach, researchers have also proposed distributed protocols [8–10] for ensuring storage correctness across multiple servers or peers.
Our work is among the first few ones in this field to consider distributed data storage in Cloud Computing. Our contribution can be summarized as the following three aspects:

1. Compared to many of its predecessors, which only provide binary results about the storage state across the distributed servers, the challenge-response protocol in our work further provides the localization of data error.
2. Unlike most prior works for ensuring remote data integrity, the new scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append.
3. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

A. Characteristics

Cloud computing is cost-effective. Here, cost is greatly reduced as initial expense and recurring expenses are much lower than traditional computing. Maintenance cost is reduced as a third party maintains everything from running the cloud to storing data. Cloud is characterized by features such as platform, location

and device independency, which make it easily adoptable for all sizes of businesses, in particular small and mid-sized. However, owing to redundancy of computer system networks and storage system cloud may not be reliable for data, but it scores well as far as security is concerned. In cloud computing, security is tremendously improved because of a superior technology security system, which is now easily available and affordable. Yet another important characteristic of cloud is scalability, which is achieved through server virtualization. Some of the most important five key characteristics are,

1. On-demand Self Service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

2. Broad Network Access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms.

3. Resource Pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data centre). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

4. Measured Service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

5. Selection of Provider

A good service provider is the key to good service. So, it is imperative to select the right service provider. One must make sure that the provider is reliable, well-reputed for their customer service and should have a proven track record in IT- related ventures. As cloud computing has taken hold, there are six major benefits that have become clear,

- Anywhere/anytime access - It promises "universal" access to high-powered computing and storage resources for anyone with a network access device.
- Collaboration among users -cloud represents an environment in which users can develop software based services and from which they can deliver them.
- Storage as a universal service - the cloud represents a remote but scalable storage resource for users anywhere and everywhere.
- Cost benefits - the cloud promises to deliver computing power and services at a lower cost.

II. Problem Statement

A. System Model

A representative network architecture for cloud data storage is illustrated in fig. 1. Three different network entities can be identified as follows:

1. User: users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
2. Cloud Service Provider (CSP): a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
3. Third Party Auditor (TPA): an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

B. The Third Party Auditor (TPA)

The Third Party Auditor (TPA), who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service security on behalf of the user upon request. Users rely on the CS for cloud data storage and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. The users may resort to TPA for ensuring the storage security of their outsourced data, while hoping to keep their data private from TPA. We consider the existence of a semi-trusted CS as does. Namely, in most of time it behaves properly and does not deviate from the prescribed protocol execution. However, during providing the cloud data storage based services, for their own benefits the CS might neglect to keep or deliberately delete rarely accessed data files which belong to ordinary cloud users. Moreover, the CS may decide to hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation. We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. TPA should be able to efficiently audit the cloud data storage without local copy of data and without bringing in additional on-line burden to cloud users.

The Cloud Computing model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called Cloud servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data. The most general forms of these operations we are considering are block update, delete, insert and append.

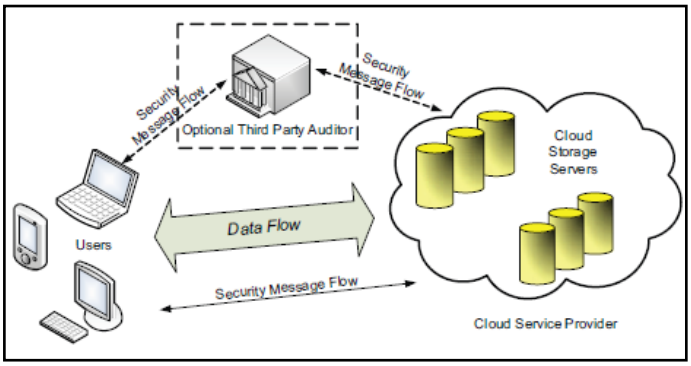

Fig. 1: Cloud Data Storage Architecture

As users no longer possess their data locally, it is of critical importance to assure users that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. In case that users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. In our model, we assume that the point-to-point
communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead. Note that we don't address the issue of data privacy in this paper, as in Cloud Computing, data privacy is orthogonal to the problem we study here.
To introduce an effective third party auditor (TPA) for privacy and security, the following fundamental requirements have to be met: TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user. The third party auditing process should bring in no new vulnerabilities towards user data privacy. They utilized and uniquely combined the public
key based homomorphic authenticator with random masking to achieve the privacy-preserving public cloud data auditing system, which meets all above requirements.
This scheme is the first to support scalable and efficient public auditing in the Cloud Computing. In particular, this scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA. The security and performance is justified through concrete experiments and comparisons with the state-of-the-art. In cloud service providers, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed
or even hiding data loss incidents so as to maintain a reputation. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the cloud architecture. Another problem is that data stored in the cloud does not remain static.

III. Proposed System

We enhance the scheme with explicit and efficient dynamic data operations for data storage security in Cloud Computing. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file while maintaining the storage correctness assurance. The straightforward and trivial way to support these operations is for user to download all the data from the cloud servers and re-compute the whole parity blocks as well as verification tokens.

A. Update Operation

In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud, from its current value fij to a new one, fij + Δ fij. We refer this operation as data update. Due to the linear property of Reed-Solomon code, a user can perform the update operation and generate the updated parity blocks by using Δ fij only, without involving any other unchanged blocks. Specifically, the user can construct a general update matrix ΔF as

$$\begin{aligned}\Delta \mathbf{F} &= \begin{pmatrix} \Delta f_{11} & \Delta f_{12} & \dots & \Delta f_{1m} \\ \Delta f_{21} & \Delta f_{22} & \dots & \Delta f_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta f_{l1} & \Delta f_{l2} & \dots & \Delta f_{lm} \end{pmatrix} \\ &= (\Delta F_1, \Delta F_2, \dots, \Delta F_m).\end{aligned}$$

Note that we use zero elements in ΔF to denote the unchanged blocks. To maintain the corresponding parity vectors as well as be consistent with the original file layout, the user can multiply ΔF by A and thus generate the update information for both the data vectors and parity vectors as follows:

$$\begin{aligned}\Delta \mathbf{F} \cdot \mathbf{A} &= (\Delta G^{(1)}, \dots, \Delta G^{(m)}, \Delta G^{(m+1)}, \dots, \Delta G^{(n)}) \\ &= (\Delta F_1, \dots, \Delta F_m, \Delta G^{(m+1)}, \dots, \Delta G^{(n)}),\end{aligned}$$

where $\Delta G^{(j)}$ ($j \in \{m+1, \dots, n\}$) denotes the update information for the parity vector $G^{(j)}$.
Because the data update operation inevitably affects some or all of the remaining verification tokens, after preparation of update information, the user has to amend those unused tokens for each vector $G^{(j)}$ to maintain the same storage correctness assurance. In other words, for all the unused tokens, the user needs to exclude every occurrence of the old data block and replace it with the new one. Thanks to the homomorphic construction of our verification token, the user can perform the token update efficiently. To give more details, suppose a block $G^{(j)}[I_s]$, which is covered by the specific token $v^{(j)}_i$, has been changed to $G^{(j)}[I_s] + \Delta G^{(j)}[I_s]$, where $I_s = \Phi_{k(i)prp}(s)$.
To maintain the usability of token v(j)i , it is not hard to verify that the user can simply update it by

$$v_i^{(j)} \leftarrow v_i^{(j)} + \alpha_i^s * \Delta G^{(j)}[I_s],$$

without retrieving any other r − 1 blocks required in the pre-computation of $v^{(j)}_i$.
After the amendment to the affected tokens, the user needs to blind the update information $\Delta g^{(j)}_i$ for each parity block in (ΔG(m+1), . . . , ΔG(n)) to hide the secret matrix P by

$$\Delta g_i^{(j)} \leftarrow \Delta g_i^{(j)} + f_{k_j}(s_{ij}^{ver}), i \in \{1, \dots, l\}.$$

Here we use a new seed sverij for the PRF. The version number ver functions like a counter which helps the user to keep track of the blind information on the specific parity blocks. After blinding, the user sends update information to the cloud servers, which perform the update operation as
G(j) ← G(j) + ΔG(j), (j ∈{1, . . . , n}).

B. Delete Operation

Sometimes, after being stored in the cloud, certain data blocks may need to be deleted. The delete operation we are considering

is a general one, in which user replaces the data block with zero or some special reserved data symbol. From this point of view, the delete operation is actually a special case of the data update operation, where the original data blocks can be replaced with zeros or some predetermined special blocks. Therefore, we can rely on the update procedure to support delete operation, i.e., by setting Δfij in ΔF to be $-\Delta fij$. Also, all the affected tokens have to be modified and the updated parity information has to be blinded using the same method specified in update operation.

C. Append Operation

The user may want to increase the size of his stored data by adding blocks at the end of the data file, which we refer as data append. We anticipate that the most frequent append operation in cloud data storage is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time. Dynamic operations are performed by constructing the matrix, where 0's indicate the blocks we need to change and 1's indicate the unchanged blocks [9].

We create a cloud environment where user, TPA and cloud server are connected each other. In public auditing system, the correctness of the data is checked by keygen, sagging, gen proof and verify proof algorithms. Homomorphism authenticator with random masking is used to achieve privacy preserving auditing scheme. The technique of bi-linear aggregate signature is used to achieve batch auditing. In cloud, the data does not remain static. We enhance the system with explicit dynamic operations in data blocks.

In some cases, the user may want to increase the size of his stored data by adding blocks at the end of the data file, which we refer as data append. We anticipate that the most frequent append operation in cloud data storage is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time. Given the file matrix F illustrated in file distribution preparation, appending blocks towards the end of a data file is equivalent to concatenate corresponding rows at the bottom of the matrix layout for file F. In the beginning, there are only l rows in the file matrix. To simplify the presentation, we suppose the user wants to append m blocks at the end of file F, denoted as ($f_{l+1,1}$, $f_{l+1,2}$, ..., $f_{l+1,m}$) (We can always use zero-padding to make a row of m elements.).With the secret matrix P, the user can directly calculate the append blocks for each parity server as

$$(f_{l+1,1}, \ldots, f_{l+1,m}) \cdot \mathbf{P} = (g_{l+1}^{(m+1)}, \ldots, g_{l+1}^{(n)}).$$

To support block append operation, we need a slight modification to our token pre-computation. Specifically, we require the user to expect the maximum size in blocks, denoted as lmax, for each of his data vector. The idea of supporting block append, which is similar as adopted in [7], relies on the initial budget for the maximum anticipated data size l_{max} in each encoded data vector as well as the system parameter $r_{max} = \lceil r * (l_{max}/l) \rceil$ for each pre-computed challenge response token. The pre-computation of the ith token on server j is modified as follows:

$$v_i = \sum_{q=1}^{r_{max}} \alpha_i^q * G^{(j)}[I_q],$$

Where

$$G^{(j)}[I_q] = \begin{cases} G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] & , [\phi_{k_{prp}^{(i)}}(q)] \leq l \\ 0 & , [\phi_{k_{prp}^{(i)}}(q)] > l \end{cases}$$

This formula guarantees that on average, there will be r indices falling into the range of existing l blocks. Because the cloud servers and the user have the agreement on the number of existing blocks in each vector $G^{(j)}$, servers will follow exactly the above procedure when re-computing the token values upon receiving user's challenge request. Now when the user is ready to append new blocks, i.e., both the file blocks and the corresponding parity blocks are generated, the total length of each vector $G^{(j)}$ will be increased and fall into the range [l, l_{max}]. Therefore, the user will update those affected tokens by adding α_i^q * $G^{(j)}[I_s]$ to the old vi

whenever $G^{(j)}[I_s] \neq 0$ for $I_s > l$, where $I_s = \phi_{k_{prp}^{(i)}}(s)$.

The parity blinding is similar as introduced in update operation, thus is omitted here.

D. Insert Operation

An insert operation to the data file refers to an append operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block F[j] corresponds to shifting all blocks starting with index j + 1 by one slot. An insert operation may affect many rows in the logical data file matrix F, and a substantial number of computations are required to renumber all the subsequent blocks as well as re-compute the challenge-response tokens. Therefore, an efficient insert operation is difficult to support and thus we leave it for our future work.

IV. Conclusion

To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. We believe that data storage security in Cloud Computing, an area full of challenges and of paramount importance, is still in its infancy now, and many research problems are yet to be identified. We envision several possible directions for future research on this area. The most promising one we believe is a model in which public verifiability is enforced. Public verifiability, supported in [6, 4], allows TPA to audit the cloud data storage without demanding users' time, feasibility or resources. An interesting question in this model is if we can construct a scheme to achieve both public verifiability and storage correctness assurance of dynamic data. Besides, along with our research on dynamic cloud data storage, we also plan to investigate the problem of fine-grained data error localization.

References

[1] Amazon.com,"Amazon Web Services (AWS)", [Online] Available: http://www.aws.amazon.com, 2008.

[2] N. Gohring,"Amazon's S3 down for several hours", [Online] Available: http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for several hours.html, 2008.

[3] A. Juels, J. Burton S. Kaliski,"PORs: Proofs of Retrievability for Large Files", Proc. of CCS '07, pp. 584–597, 2007.
[4] H. Shacham, B. Waters,"Compact Proofs of Retriev-ability", Proc. of Asiacrypt '08, Dec. 2008
[5] M.A.Shah, R. Swaminathan, M. Baker,"Privacypreserving audit and extraction of digital contents", Cryptology ePrint Archive, 2008.
[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, D. Song,"Provable Data Possession at Untrusted Stores", Proc. Of CCS '07, pp. 598–609, 2007.
[7] G. Ateniese, R. D. Pietro, L. V. Mancini, G. Tsudik, "Scalable and Efficient Provable Data Possession", Proc. of SecureComm '08, pp. 1–10, 2008.
[8] T. S. J. Schwarz, E. L. Miller,"Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage", Proc. of ICDCS '06, pp. 12–12, 2006.
[9] Wang, K. Ren, W. Lou,"Achieving secure, scalable, and fine-grained access control in cloud computing", in Proc. of IEEE INFOCOM'10, San Diego, CA, USA, 2010.
[10] K. D. Bowers, A. Juels, A. Oprea,"HAIL: A High-Availability and Integrity Layer for Cloud Storage", Cryptology ePrint Archive, Report 2008/489, 2008, [Online] Available: http://eprint.iacr.org/.

Mr Moshe dayan.Sirapangi is a final year student of KIET-Kakinada Institute of Engg and Tech,Korangi. Presently He is pursuing his M.Tech (C.S) from this college and he received his Bachelor of Technology (B.Tech) from JNTUK University, His area of interest includes Computer Networks and Object oriented. Programming languages in Computer Applications.

Mr G.B.V.Padmanadh, an Efficient teacher, received M.Tech in the stream of Computer Science & Technology from Githam university, Visakhapatnam, and is been persuing Ph.D from JNTU Kakinada and He is working as an Associate Professor in Department of C.S,Kakinada Institute of Engg. and Tech, Korangi. He has 8 years of teaching experience. He has published many papers in both National & International Journals. His area of Interest includes Data Communications & Networks, Database Management Systems, C Programming and other advances in Computer Applications.