ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 4 (Oct. - Dec. 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Trustable Data Storage Services in Cloud Computing

[I]Pradeep P, [II]K. Shirisha Reddy
[I]M.Tech (CSE), [II]Associate Professor Dept. of CSE
[I,II]Vignana Bharathi Institute of Technology, Hyderabad, India

Abstarct
Cloud Computing provides new vision to the world. The approach of cloud computing is totally different from traditional system. In traditional system for any service, we need purchase, install, maintain, update by own. But in cloud environment we just pay for that service according to our usage basis & here no need to worry about purchase, install, maintenance and update. It is beneficial with respect to cost, flexibility, scalability. We require only good bandwidth network for better performance. Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, such a service is also relinquishing users' physical possession of their outsourced data, which inevitably poses new security risks towards the correctness of the data in cloud. In order to address this new problem and further achieve a secure and dependable cloud storage service, we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure-coded data. The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

Keywords
Cloud Computing, TPA, Data Integrity, Token, Dynamic Operations.

I. Introduction

Cloud computing is the term used to share the resources globally with less cost .we can also called as 'IT ON DEMAND'. It provides three types of services I.e., Infrastructure as a service (IaaS) ,Platform as a service(PaaS) and Software as a service(SaaS). End users access the cloud based applications through the web browsers with internet connection. Moving data to clouds makes more convenient and reduce to manage hardware complexities. Data stored at clouds are maintained by Cloud service providers (CSP) with various incentives for different levels of services. However it eliminates the responsibility of local machines to maintain data, there is a chance to lost data or it effects from external or Internal attacks. To maintain the data integrity and data Availability many people proposed several algorithms and Methods that enable on demand data correctness and verification. So Cloud servers are not only used to store data like a ware house , it also provides frequent updates on data by the users with different operations like insert, delete , update and append. Lastly the deployment of cloud computing is powered by the data centers running in cooperated and distributed manner Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works under different systems and security models. Researchers also proposed distributed protocols to ensure storage correctness across multiple servers. In this paper we mainly focus on dynamic generation of tokens for verification of blocks and token precipitation before distributing the files in to cloud, the rest of the operations are taken from the same paper(towards secure and dependable storage services in cloud computing). We use homomorphic token for challenge responses with few parameters. It will easily trace the misbehaving servers which are having corrupted data blocks. Wended not concern about distribution of blocks we can distribute the blocks redundantly to many servers or disperse the blocks to different servers with server persistent schemes. We also provide the extension of proposed scheme to support third –party auditing, where user can safely delegate the integrity checking task to trusted third party providers.

II. Problem Statement

1. System Model

The cloud storage system architecture consists of following network entities User: An entity, who performs data storage and retrieval operations without knowing the internalissues.Cloud Server (CS): An entity, which provides data storage space and resources required for computations, cloud servers are managed by cloud service providers. Third Party Auditor (TPA): An optional Entity, but here we use TPA as Trusted party and to perform some computations instead of users. In cloud data storage system, user can upload or stores the data into cloud or use services from the cloud (Here we focused on file storage and retrieval operations). user stores data into set of cloud servers which are running in a distributed and cooperated manner. Data redundant techniques can be employed using erasure correcting code to protect from faults or server crashes. Users can perform manipulations on stored data like insert update and append through blocks. Block level updating and deletions are allowed with token checking. If user has not having enough resources to compute tokens or required hardware support then he can easily delegate the work to a third party auditor called as TPA. He is responsible to generate homomorphic token and stores the token persistently and securely for further verification. In our scheme we assume that TPA is secure and he is responsible to protect from threats, users will pay some incentives to TPA for maintenance.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

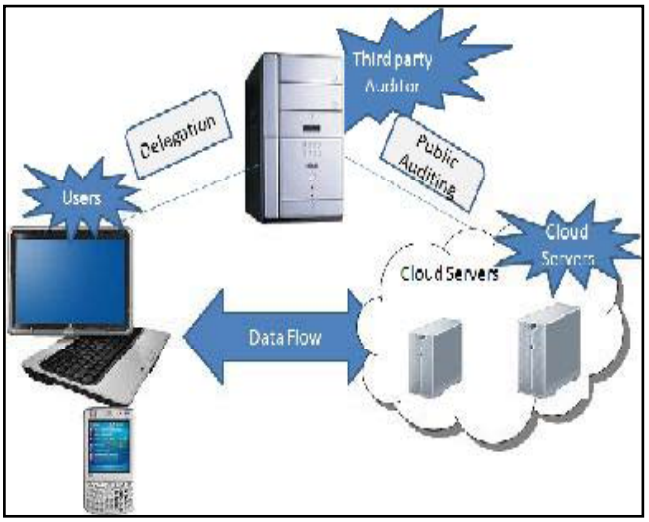

Fig 1: Cloud Storage System Architecture

2. Adversary Model

Adversary model was introduced to explore some of threats associated in this model. As we know that the data is not present at users place because data is stored at cloud servers. It may lead to some security threats mainly two, internal attacks and external attacks. Internal attacks comes from the cloud servers itself, these servers may be malicious and lead to byzantine failures and hide some data loss issues.

Secondly external attacks are from outsiders who are compromised the data from cloud service providers without its permission. Outsider attacks may lead to modification of data or deleting the users and so on which are completely masked from cloud service providers. All though TPA can also possibly hack the data for itself interested and it is also a case for inside attacks, but we ensure that TPA's are trusted party servers. Therefore, we consider the adversary in our model to capture all types of attacks both internal and external threats. Once the server is compromised, the data is polluted with fraudulent data and users cannot get the original data from the clouds.

3. Design Goals

Our main goal is to ensure the data integrity and security .In this paper, we aim to design

- Precipitation token key generation algorithm which is simple, elegant and secures method and less overhead due to few parameters that has to be chosen.
- Challenge verification scheme was designed in easy and efficient way to prevent data from byzantine server failures and data dependability detection or detect data errors on blocks.
- Cloud servers ensure that the file was saved successfully without block modifications. This can be achieved by two way token checking.

III. Ensuring Data Storage Over Cloud

In cloud data storage system, users store their data remotely i,e., on clouds, so that the correctness and availability of data files must be guaranteed to be identical. Our aim is to detect the servers which behaves differently and may leads to internal and external threats. In this paper, we explore the technique used to detect the modified blocks easily with very less overhead using homomorphic token precipitation technique ,later we can use erasure coded technique to acquire the desired blocks from different servers.

1. File Distribution Preparation

It is well known that rate less erasure-correcting code (raptor code) may be used to tolerate multiple failures in distributed storage systems. In cloud data storage, we rely on this technique to disperse the data file F redundantly across a set of n= m+k distributed servers. Raptor codes pre-encode the source symbols using a fixed length block code, and then encode these new symbols with an LT code. The main advantage is that, for correct decoding, it is no longer necessary that the LT decoding succeeds for all the symbols. Thus, it is possible to use a simpler degree distribution that does not recover all the symbols but makes the decoding process faster. The decoding algorithm is composed of two steps. The inner LT decoder returns a hard bit-reliability vector. This latter is processed by the outer LDPC decoder, based on belief propagation algorithm.

2. Challenge Token Precomputation

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) ($j \in \{1, \ldots, n\}$), each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens pre-computed by the user. In our case here, the user stores them locally to obviate the need for encryption and ower the bandwidth overhead during dynamic data operation.

3. Correctness Verification and Error Localization

Error localization is a key prerequisite for eliminating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization, thus only provide binary results for the storage verification. Our scheme outperforms those by integrating the correctness verification and error localization in our challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s). Once the inconsistency among the storage has been successfully detected, we can rely on the pre-computed verification tokens to further determine where the potential data error(s) lies and give details of correctness verification and error localization.

4. File Retrieval and Error Recovery

Since our layout of file matrix is systematic, the user can reconstruct the original file by downloading the data vectors from the first m servers, assuming that they return the correct response values. Notice that our verification scheme is based on random spot-checking, so the storage correctness assurance is a probabilistic one. However, by choosing system parameters (e.g., r,l,t) appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability. On the other hand, whenever the data corruption is detected, the comparison of pre-computed tokens and received response

values can guarantee the identification of misbehaving server(s), again with high probability. Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by rate less erasure correction code. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

5. Towards Third Party Auditing

As discussed in our architecture, in case the user does not have the time, feasibility or resources to perform the storage correctness verification, he can optionally delegate this task to an independent third party auditor, making the cloud storage publicly verifiable. However, as pointed out by the recent work, to securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy. Namely, TPA should not learn user's data content through the delegated data auditing. Now we show that with only slight modification, our protocol can support privacy-preserving third party auditing. The new design is based on the observation of linear property of the parity vector blinding process. Recall that the reason of blinding process is for protection of the secret matrix P against cloud servers. However, this can be achieved either by blinding the parity vector k or by blinding the data vector m (we assume $k < m$). Thus, if we blind data vector before file distribution encoding, then the storage verification task can be successfully delegated to third party auditing in a privacy-preserving manner.

IV. Dynamic Data Operation Support

In this section, we will show how our scheme can explicitly and efficiently handle dynamic data operations such as update, delete, append and insert operations for cloud data storage, by utilizing the linear property of Reed-Solomon code and verification token construction.

1. Update Operations

Due to the linear property of Reed-Solomon code, a user can perform the update operation and generate the updated parity blocks by using Δfij only, without involving any other unchanged blocks. 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.

2. Delete Operations

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.

3. Append Operations

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.

4. Insert Operations

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 corresponds to shifting all blocks starting with index by one slot. Thus, an insert operation may affect many rows in the logical data file matrix, and a substantial number of Computations are required to renumber all the subsequent blocks as well as recomputed the challenge-response tokens.

V. Related Discussions

In this paper, we mainly identify the algorithm for token generation in a different way with user parameters. Here this algorithm also results in a good performance and we just cover the topics in brief. Many algorithms and many approaches are there to perform computations. But our scheme is provided with very less over head. If there is any change in the block, the block will not give the same token value. Here we are encrypting two times firstly with user parameters and Token key is called as file Token for entire file and finally follows bit permutation approach. Homomorphic technique ensures that we need not decrypt the key for data checking instead we can compare directly with encrypted token. The file Token is always present for each file of each user so there will be absolutely no collisions among different users of the same file. We were not mentioned split function here but this function gives the key in very less size with effective mathematical calculations. The secret matrix which we were considered contains the set of special symbols and this vector is randomized for every file.

VI. Conclusion

Cloud computing is a new paradigm in which users can store their data. However, security and privacy issues impose strong barrier for users' adoption of Cloud systems and Cloud services. In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support. We rely on rate less 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 rate less 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). Public verifiability, supported in allows TPA to audit the cloud data storage without demanding users' time, feasibility or resources.

References

[1]. *Cong Wang; Qian Wang; Kui Ren; Ning CaoWenjing Lou; "Toward Secure and Dependable Storage Services in Cloud Computing," Services Computing, IEEE Transactions on , vol.5, no.2,pp.220-232, April-June 2012*

[2]. *C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in Proc. of IWQoS'09, July 2009, pp. 1–9.*

[3]. *"Auditing to keep online storage service honest "by M.A. Shah, R Swaminathan.*

[4]. *M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc.*

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07), pp. 1–6, 2007.

[5]. *Amazon online shopping web services url link : http://aws.amazon.com*

Authore Profile

Pradeep P, Pursuing M.Tech (CSE) from Vignana Bharathi Institute of Technology, Hyderbad, India.