IPHV2I10001X
International Journal Of Advanced Research and Innovation -Vol.II, Issue .2
ISSN Online: 2319 – 9253
Print: 2319 – 9245

Effective and Secure Data Storage in Cloud Computing

N. SARITHA[1] A. RAJASHEKAR REDDY[2]

1. [M.Tech (CS)] , Dept. of Computer Science & Engineering, VITS(N6), Karimnagar.
2. Assistant Professor, Dept. of Computer Science & Engineering, VITS(N6), Karimnagar.

ABSTRACT

Cloud service utilization are increasing gradually. Web servers are now a day's contents of its are outsourced to cloud servers due to the maintenance cost of storage devices and peripherals. Considerable issue is that security why because the cloud servers are data centers maintains the large volumes of data. So we suppose third party auditor on behalf of the cloud client to provide protection against various attacks. TPA performs multiple auditing tasks simultaneously. Which improves not only backup or archive and also ensures integrity. TPA performs multiple auditing tasks simultaneously and achieves security.

INTRODUCTION

Cloud computing is a term used to describe both a platform and type of application. As a platform it supplies, configures, reconfigures the servers while the servers can be physical machines or virtual machines. On the other hand cloud computing describes applications that are extended to be accessible through the internet and for this purpose large data centers and powerful servers are used to host the web applications and web services.

Benefits of cloud computing

The benefits of cloud computing are Reduced Data Leakage, Decrease evidence acquisition time, they eliminate or reduce Service downtime, they Forensic readiness, they decrease evidence transfer time.

Drawbacks of cloud computing

Few of the disadvantages associated with cloud computing are:High Speed Internet Required Constant Internet Connection

Batch auditing

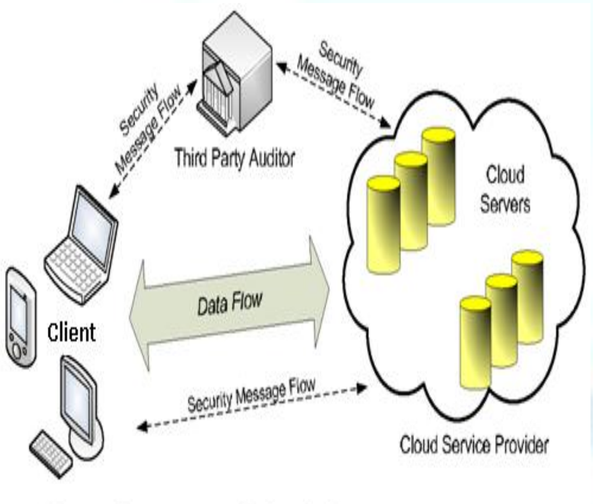
There are K users having K files on the same cloud. They have the same TPA Then, the TPA can combine their queries and save in computation time

The comparison function that compares the aggregate authenticators has a property that allows checking multiple messages in one equation Instead of 2K operation, K+1 are possible

Data dynamics

The data on the cloud may change according to applications This is achieved by using the data structure Merkle Hash Tree (MHT) With MHT, data changes in a certain way; new data is added in some places There is more overhead involved user sends the tree root to TPA

Architecture



The architecture of cloud data storage service

Existing System

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.

Disadvantages

Especially to support block insertion, which is missing in most existing schemes.

Proposed Model

Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations.

Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data.

Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

Advantages

1) We motivate the public auditing system of data storage security in Cloud Computing, and propose a protocol supporting for fully dynamic data operations, especially to support block

insertion, which is missing in most existing schemes;

2) We extend our scheme to support scalable and efficient public auditing in Cloud Computing. In particular, our scheme achieves auditing tasks from different users can be performed simultaneously by the TPA.

3) We prove the security of our proposed construction and justify the performance of our scheme through concrete implementation and comparisons.

Approaches

1. **Public auditability for storage correctness assurance**

To allow anyone, the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.

2. **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 auditability and dynamic data operation support.

3. **Blockless verification**

No challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

4. **Dynamic Data Operation with Integrity Assurance**

Now we show how our scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. Note that in the following descriptions, we assume that the file F and the signature _ have already been generated and properly stored at server. The root metadata R has been signed by the client and stored at the cloud server, so that anyone who has the client's public key can challenge the correctness of data storage.

5. **Data Modification:**

We start from data modification, which is one of the most

frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones. At start, based on the new block the client generates the corresponding signature. The client signs the new root metadata R′ by sigsk(H(R′)) and sends it to the server for update. Finally, the client executes the default integrity verification protocol. If the
Output is TRUE, delete sigsk(H(R′)),and generate duplicate file.

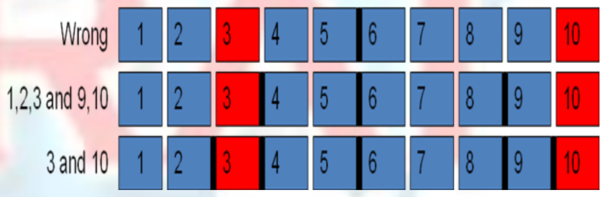
6. Batch Auditing for Multi-client Data:

As cloud servers may concurrently handle multiple verification sessions from different clients, given K signatures on K distinct data files from K clients, it is more advantageous to aggregate all these signatures into a single short one and verify it at one time. To achieve this goal, we
extend our scheme to allow for provable data updates and verification in a multi-client system.

The signature scheme allows the creation of signatures on arbitrary distinct messages. Moreover, it supports the aggregation of multiple signatures by distinct signers on distinct messages into a single short signature, and thus greatly reduces the communication cost while providing efficient verification for the authenticity of all messages.

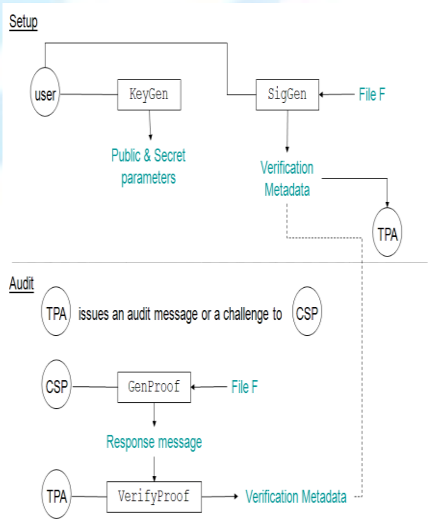
Performance with Invalid Responses

- In batch auditing, true means that all of the messages are correct
- False means at least one is wrong
  - Divide batch in half, repeat for left- and right parts
  - Binary search



<u>Algorithm Techniques:</u>

- Setup Phase
- Audit Phase



The client's public key and private key are generated by invoking KeyGen(·). By running SigGen(·), the data file F is pre-processed, and the homomorphic authenticators together with metadata are produced.

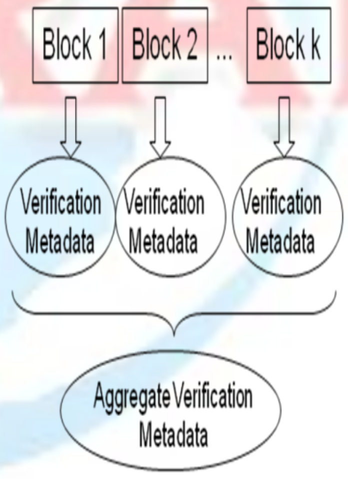
KeyGen(1k). The client generates a random signing key pair (spk, ssk). Choose a random α ← Zp and compute v ← gα. The secret key is sk = (α, ssk) and the public key is pk = (v, spk). SigGen(sk, F). Given F = (m1,m2 . . . ,mn), the client chooses a random element u ← G. Let t = name||n||u||SSigssk(name||n||u) be the file tag for F.

Then the client computes signature σi for each block mi (i = 1, 2, . . . , n) as σi ← (H(mi) · umi)α. Denote the set of signatures by _ = {σi}, 1 ≤ i ≤ n. The client then generates a root R based on the construction (pk, sk) ← KeyGen(1k). This probabilistic algorithm is run by the client. It takes as input security parameter 1k, and returns public key pk and private key sk.(_, sigsk(H(R))) ← SigGen(sk, F).

This algorithm is run by the client. It takes as input private key sk and a file F which is an ordered collection of blocks {mi}, and outputs the signature set _, which is an ordered collection of signatures {σi} on {mi}. It also outputs metadata-the signature sigsk(H(R)) of the root R of a Merkle hash tree. In our construction, the leaf nodes of the hashes of H(mi). (P) ← GenProof(F,_, chal). This algorithm is run by the server. It takes as input a file F, its signatures _, and a challenge chal. It outputs a data integrity proof P for the blocks specified by chal.

Privacy-Preserving Public Auditing Scheme

Uses homomorphic authenticator Also uses a random mask achieved by a Pseudo Random Function (PRF)



A linear combination of data blocks can be verified by looking only at the aggregated authenticator

CONCLUSION

Finally we conclude that the proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. 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. 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). The most promising one we believe is a model in which public verifiability is enforced. Public verifiability, supported in allows TPA to audit the cloud data storage without demanding users' time, feasibility or resources.

REFERENCES

[1] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws. amazon.com,2008. N. Gohring, "Amazon's S3 down for several hours," Online.

[2]At http://www.pcworld.com/businesscenter/article/142549/amazo s s3 down for several hours.html, 2008.

[3] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, pp. 584–597, 2007. H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. of Asiacrypt '08, Dec. 2008.

[4] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. Of CCS '07, pp. 598– 609, 2007.

[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. of SecureComm '08, pp. 1– 10, 2008.

[7] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: UsingAlgebraic Signatures to Check Remotely Administered Storage," Proc.