RESEARCH ARTICLE OPEN ACCESS

Efficient Implementation of Proof of Retrievability (OPOR) In Cloud Computing With Resource Constrained Devices

Shalini J and Dr. K. Raghuveer
Department of Information Science and Engineering, N.I.E Mysore
Karnataka, India

ABSTRACT
Cloud computing has become an integral part of IT services, storing the application softwares and databases in large centralized shared data servers. Since it's a shared platform, the data and services may not be fully trust worthy. In this work, we have implemented an efficient security model that ensures the data integrity of stored data in cloud servers. The computational load of data verification linearly grows with the complexity of the security model and this poses a serious problem at the resource constrained user's end. Therefore to tackle this problem we have implemented a new cloud storage scheme which ensures proof of retrivebility (OPoR) at a third party cloud audit server to pre-process data before uploading into cloud storage server.

I. INTRODUCTION

Cloud Computing has been envisioned as the next generation architecture of the IT enterprise due to its long list of unprecedented advantages: on-demand self- service, ubiquitous network access, location-independent resource pooling, rapid resource elasticity[5, 6, 7, 10], and usage- based pricing. In particular, 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.

Although having appealing advantages as a promising service platform for the Internet, this new data storage paradigm in "Cloud" brings many challenging issues which have profound influence on the usability, reliability, scalability, security, and performance of the overall system. One of the biggest concerns with remote data storage is that of data integrity verification at untrusted servers. For instance, the storage service provider may decide to hide such data loss incidents as the Byzantine failure from the clients to maintain a reputation. What is more serious is that for saving money and storage space the service provider might deliberately discard rarely accessed data files which belong to an ordinary client. Considering the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verification without the local copy of data files.

Cloud Computing moves the application soft- ware and databases to the centralized large data centers [11, 14, 16, 19], where the management of the data and services may not be fully trustworthy. In this work, we study the problem of ensuring the integrity of data storage in Cloud Computing. To reduce the computational cost at user side during the integrity verification of their data, the notion of public verifiability has been proposed. However, the challenge is that the computational burden is too huge for the users with resource-constrained devices to compute the public authentication tags of file blocks. To tackle the challenge, we propose OPoR, a new cloud storage scheme involving a cloud storage server and a cloud audit server, where the latter is assumed to be semi-honest.

II. RELATED WORK

Provable Data Possesion:

In paper [1] author introduced a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems. We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

Author focused on the problem of verifying if an untrusted server stores a client's data. Our solutions for PDP fit this model: They incur a low (or even constant) overhead at the server and require a small, constant amount of communication per challenge. Key components of our schemes are the homomorphic verifiable tags.

Proofs of retrievability:

In paper [2], authors define and explore proofs of retrievability (PORs). A POR scheme enables an archive or back-up service (prover) to produce a concise proof that a user (verifier) can retrieve a target file F, that is, that the archive retains and reliably transmits file data sufficient for the user to recover F in its entirety. A POR may be viewed as a kind of cryptographic proof of knowledge (POK), but one specially designed to handle a large file (or bit string) F. We explore POR protocols here in which the communication costs, number of memory accesses for the prover, and storage requirements of the user (verifier) are small parameters essentially independent of the length of F. In addition to proposing new, practical POR constructions, we explore implementation considerations and optimizations that bear on previously explored, related schemes. In a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of F. PORs give rise to a new and unusual security definition whose formulation is another contribution of our work. We view PORs as an important tool for semi-trusted online archives. Existing cryptographic techniques help users ensure the privacy and integrity of files they retrieve. It is also natural, however, for users to want to verify that archives do not delete or modify files prior to retrieval. The goal of a POR is to accomplish these checks without users having to download the files themselves. A POR can also provide quality-of-service guarantees, i.e., show that a file is retrievable within a certain time bound.

In paper [12], authors proposed a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient. Our preliminary experiment conducted on Amazon EC2 instance further demonstrates the fast performance of the design.

Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources.

Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a third party auditor (TPA) to check the integrity of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user.

Dynamic audit services for integrity veri cation:

In paper [13], authors proposed a dynamic audit service for verifying the integrity of untrusted and outsourced storage. Our audit service, constructed based on the techniques, fragment structure, random sampling and index-hash table, can support provable updates to outsourced data, and timely abnormal detection. In addition, we propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services. Our experimental results not only validate the effectiveness of our approaches, but also show our audit system has a lower computation overhead, as well as a shorter extra storage for audit metadata.

In this work, we introduce a dynamic audit service for integrity verification of untrusted and outsourced storages. Our audit system, based on a novel audit system architecture, can support dynamic data operations and timely abnormal detection with the help of several effective techniques, such as fragment structure, random sampling, and index-hash table. Furthermore, we propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services. A proof of concept prototype is also implemented to evaluate the feasibility and viability of our proposed approaches. Our experimental results not only validate the effectiveness of our approaches, but also show our system has a lower computation cost, as well as a shorter extra storage for integrity verification.

Data Dynamics for Storage Security:

Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise [24]. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing.

The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step to- ward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public verifiability or dynamic data operations, this paper achieves both. We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for seamless integration of these two salient features in our protocol design. In particular, to achieve efficient data dynamics, we improve the Proof of Retrievability model [1] by manipulating the classic Merkle Hash Tree (MHT) construction for block tag authentication. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

III. PROBLEM DEFINITION

Data storage paradigm in “Cloud” brings many challenging issues which have profound influence on the usability, reliability, scalability, security, and performance of the overall system.

One of the biggest concerns with remote data storage is that of data integrity verification at untrusted servers.

For instance, the storage service provider may decide to hide such data loss incidents as the Byzantine failure from the clients to maintain a reputation. What is more serious is that for saving money and storage space the service provider might deliberately discard rarely accessed data files which belong to an ordinary client.

Considering the large size of the outsourced electronic data and the client’s constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verification without the local copy of data files.

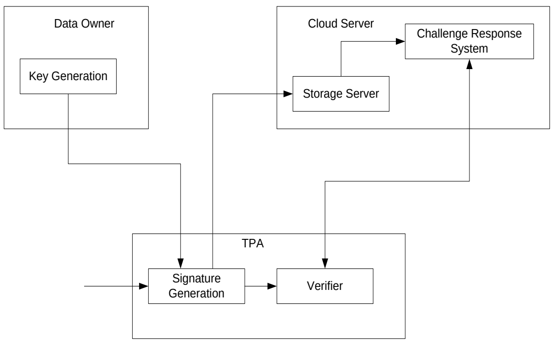
IV. Proposed Solution

We present an efficient verification scheme for ensuring remote data integrity in cloud storage. The proposed scheme is proved secure against reset attacks in the strengthened security model while supporting efficient public verifiability and dynamic data operations simultaneously proposed a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and do not support fully dynamic data operations. In [22], Wang et al. considered dynamic data storage in distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [11], they only considered partial support for dynamic data operation. In [21], they also considered how to save storage space by introducing reduplication in cloud storage. Recently, Zhu et al. [19] introduced the provable data possession problem in cooperative cloud service providers and designed a new remote integrity checking system.

Our proposed method consists of following stages:
1. Learning and Analysis phase.
2. Design and Implementation phase.
3. Testing Phase.

System Architecture

System architecture is the conceptual design that defines the structure and behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.



The system has 3 sub systems:

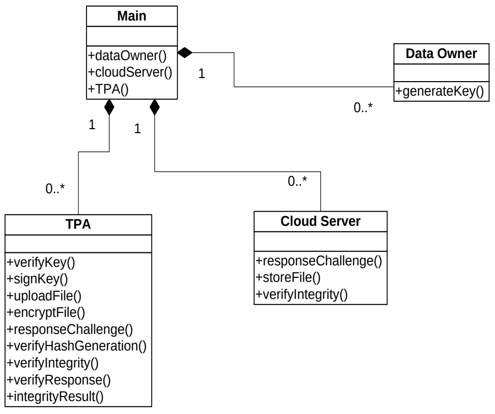
Data Owner: This module will implement the functionality of generating the key for encrypting the file.

Cloud Server: It stores the files in the cloud & responds to integrity challenge request from the TPA.

TPA: Signed hash content is been sent. Encrypted file is sent to cloud server. TPA verifies the integrity by posing challenge request to the cloud server & then checks the validity. It raises alert to the data owner if the integrity has failed.

Classes Designed for the system
A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that

describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.



The class diagram has the following classes
Main class: This class has operations called data owner, cloud server and TPA.
Data Owner: This class has operations called generate Key.
TPA: This class has operations called verify Key, sign key, upload file, encrypt file, response challenge, verify hash generation, verify integrity and verify response, integrity result.
Cloud Server: This class has operations called response challenge, store file and verify integrity.
Basic Structure of Authentication Scheme Employed in our Security Model
Data Integrity- Assumptions:
1) Mechanism in place to securely share data between Data Owner and Clients.
2) The data could be the public key of the DO or collision resistant hash data.

The conventional data authenticity verification poses huge computation overload, because it has an exponential storage overhead as distinct signatures needs to store with each tuple. There are two integrity schemes- Probabilistic and Deterministic. In our implementation we follow deterministic approach that is generally based on Authenticated Data Structures (ADS). It's a technique in which some kind of authentication data is stored on the DSP. On the client's query, a DSP returns the queried data along with some extra authentication data that is then used by the client to verify the authenticity of returned data.

Merkel Hash Tree:
Security of this signature scheme depends on the security of the hash function.
• Only one hash needs to be maintained or shared securely.
• To authenticate any data block only log2 n hashes need to be transferred, where n denotes total number of data blocks.
• In case of integrity checking of a continuous range of blocks, even less than log2 n hashes need to be transferred.

$$H(d) = \begin{cases} h(h(d.val)||h(d.name)) \\ h(h(d.content)||h(d.tagname)) \\ H(child(1,d)||\ldots||H(child(N,d)) \end{cases}$$

Where "||" denotes the concatenation function.

V. RESULTS

The key contributions of our project are efficient public verifiability with enhanced security model, while preventing reset attacks and fast, simultaneous bidirectional dynamic data operations. This is achieved by outsourcing all the computationally expensive tasks like data verification, security authentication from client end to a third party Cloud Audit Service (CAS).

Steps implemented in our scheme:
1) Setup phase: It takes as input security parameter (I^k) and returns public parameter (p^k) and private parameter (s^k).

2) Data Upload phase: there are two sub-stages in this algorithm

➢ Client uploads the data file (F) to a Cloud Audit Server (CAS), where F is an ordered collection of blocks $\{M_i\}$ and is encoded using rate-ρ error correcting codes.

➢ In the second stage the data file (F) is re-uploaded to the Cloud Storage Service (CSS), inputs for this function are $(s^k,\ F)$ and it outputs set of signature set (ϕ) for F, which is ordered collection of signature $\{\sigma_i\}$ on $\{M_i\}$. Where, $\phi = [\sigma_1 \quad \sigma_2 \quad \sigma_i]$ and the stored file on CSS is now denoted as $F^{\star} = \{\ F\ ,\phi\}$. It also outputs the metadata the root of R of a Merkel hash tree as shown in (Figure 1) and the authentication tag $t = sig_{sk} = (h(R))$ of $F^{\star}$.

3) Data Integrity Verification Phase: This is an interactive protocol for verifying the integrity of the uploaded data on the audit cloud storage. Successful retrievability of any data stored on cloud depends on the integrity verification scheme. The CSS has to provide the proof of retrievability, $P(p^k, F^{\star}, t\ )$ and CAS plays the role of verifier, $V(p^k, t\ )$. At the end of the protocol, $V(p^k, t\ ) = \begin{cases} 1\ if\ True \\ 0\ if\ False \end{cases}$. The output of this phase will be either 1 or 0 depending on the result of verification.

4) Dynamic Data Update Phase: This is an additional feature of our algorithm; it allows the user to dynamically update the data files on cloud server.

The uploaded data $F^{\star}$on the CSS could be dynamically modulated to $\hat{F}^{\star}$ with tag $\tilde{t}$. The CAS plays the role of the verifier with input, the private key s^k, $\tilde{t}$ and an operation request "update" from the client. At the end of the protocol, V outputs a file tag of the updated file if CSS provides a valid proof for the update.

Correctness of the bidirectional data integration and security is verified by the following two steps:

➢ If $(F^{\star}, t) \backsimeq Upload\,(s^k, F)$, then Integrity verify $\{P(p^k, F^{\star}, t) \backsimeq V(p^k, \mathrm{t})\} = 1$.

➢ If$(F^{\star}, t) \backsimeq Update\,\{P(p^k, F^{\star}, t) \backsimeq$ *Vsk*,t, update, then Integrity verify *Ppk,F⋆,t* ≃*Vpk*,t*=1.*

The computational load of the above is determined by the following two steps:

$$\sigma_i = \left(H(M_i) \cdot \prod_{j=1}^{s} {u_j}^{M_{ij}} \right)^{\alpha}$$

$$\mu_j = \sum_{(i,v)} v_i \cdot M_{ij}$$

$$e(\sigma, g) = e\left(\prod_{(i,v)} H(M_i)^v \cdot \prod_{j=1}^{s} {u_j}^{\mu_j}, v \right)^{\alpha}$$

The computational load analysis is shown in Fig 1. Our efficient implementation of the algorithm ensures that the computational time of audit compute and verification doesn't linearly increase with the size of the data file. The distribution is rather sigmoidal and saturates for larger file sizes, demonstrating the feasibility of our algorithm.

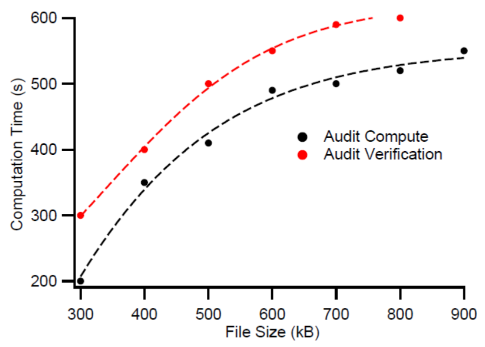


Figure 1: Computational load analysis

$$y = k_0 + \frac{k_1}{\left(1 + \exp\left(-\frac{(x - k_2)}{k_3}\right)\right)}$$

$y = Computation\ time,\ x = File\ size$

$k_0 = 60.67,\ k_1 = 564.2,\ k_2 = 360.42,\ k_3 = 132.15$

VI. CONCLUSION

This paper proposes OPoR, a new proof of retrievability for cloud storage, in which a trustworthy audit server is introduced to pre-process and upload the data on behalf of the clients. In OPoR, the computation overhead for tag generation on the client side is reduced significantly. The cloud audit server also performs the data integrity verification or updating the outsourced data upon the clients' request. Besides, we construct another new PoR scheme proven secure under a PoR model with enhanced security against reset attack in the upload phase. The scheme also supports public variability and dynamic data operation simultaneously.

REFERENCES

[1]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2007, pp. 598–609.

[2]. A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2007, pp. 584–597.

[3]. H. Shacham and B. Waters, "Compact proofs of retrievabil-ity," in *ASIACRYPT '08: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 90–107.

[4]. K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: theory and implementation," in *Proceedings of CCSW 2009*. ACM, 2009, pp. 43–54.

[5]. M. Naor and G. N. Rothblum, "The complexity of online memory checking," *J. ACM*, vol. 56, no. 1, pp. 2:1–2:46, Feb. 2009. [Online]. Available: http://doi.acm.org/10.1145/1462153. 1462155

[6]. E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in *Proceedings of ESORICS 2008, volume 5283 of LNCS*. Springer-Verlag, 2008, pp. 223–237.

[7]. M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008, http://eprint.iacr.org/.

[8]. A. Oprea, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in *In Proc. of NDSS 2005*, 2005.

[9]. T. S. J. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2006.

[10]. L. V. M. Giuseppe Ateniese, Roberto Di Pietro and G. Tsudik, "Scalable and efficient provable data possession," in *International Conference on Security and Privacy in Communication Networks (SecureComm 2008)*, 2008.

[11]. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM*, 2010, pp. 525–533.

[12]. Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *SAC*, 2011, pp. 1550–1557.

[13]. Q. Zheng and S. Xu, "Fair and dynamic proofs of retrievability," in *CODASPY*, 2011, pp. 237–248.

[14]. J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on attribute-based encryption," *ES-ORICS*, 2013.

[15]. J. Li and K. Kim, "Hidden attribute-based signatures without anonymity revocation,"

[24]. 2013, pp. vol. 1, no. 1.

Information Sciences, vol. 180, no. 9, pp. 1681–1689, 2010.

[16]. J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," *ICICS*, 2012.

[17]. X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms of outsourcing modular exponentiations," *ESORICS*, pp. 541–556, 2012.

[18]. Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, 2012.

[19]. H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, "Towards end-to-end secure content storage and delivery with public cloud," in *CODASPY*, 2012, pp. 257–266.

[20]. Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *CODASPY*, 2012, pp. 1–12.

[21]. C. Wang, Q. Wang, and K. Ren, "Ensuring data storage security in cloud computing," in *Proceedings of IWQoS 2009*, Charleston, South Carolina, USA, 2009.

[22]. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dy-namic provable data possession," Cryptology ePrint Archive, Report 2008/432, 2008, http://eprint.iacr.org/.

[23]. X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud,," in *IEEE Transactions on Cloud Computing*,