Theory and Implementation of outsourced Proof of Retrievability in Cloud

K.Anuja
PG Scholar
Department of Computer Science & Engineering
Christian College of Engineering and Technology
Dindigul, Tamilnadu-624619, India

Dr.A.Nirmal Kumar
Assistant Professor
Department of Computer Science & Engineering
Christian College of Engineering and Technology
Dindigul, Tamilnadu-624619, India

Abstract

To ensure the data integrity of the outsourced data Proof-of-Retrievability (POR) schemes has been proposed wherein storage server must prove to a verifier that all of a client's data are stored correctly. Existing PoR schemes have computational complexities and allow only private verification. We focus on designing a scheme that achieves both public verifiability and constant communication cost simultaneously,and releases the data owners from the burden of staying online.In our proposed scheme, the message exchanged between the prover and verifier is composed of a constant number of group elements. We combine techniques such as constant size polynomial commitment and homomorphic linear authenticators.The Computational Diffie-Hellman Problem, the Strong Diffie-Hellman assumption and the Bilinear Strong Diffie-Hellman assumption is used to prove the security of our POR Scheme.

Keywords: Public Verifiability, Proof of Retrievability, Computational Complexities, Prover, Verifier, Data Integrity

I. INTRODUCTION

Data outsourcing is the main benefit of cloud computing. A third party cloud storage server authentically stocks up the data with it and offer it back to the client whenever required. The storage of data needs the hardware to be updated frequently in order to give space to the large volume of data.Storage outsourcing reduces the cost of storage, maintenance also assure a reliable storage of important data by keeping multiplecopies of the data thereby reducing the chance of losing data by hardware failures.

Assurance of data availability and download ability to end-users is assured with proof of Retrievability scheme and also guarantees that the data has not undergone any alterations. A POR is a protocol in which a server/archive proves to a client that a target file F is intact, in thesense that the client can retrieve all of F from the server with high probability and make use of a challenge-response format with low communication complexity.Shacham and Waters (SW) offer an alternative construction based on the idea of storing homomorphic block integrity values that can be aggregated to reduce the communication complexity of a proof. Its main advantage is that, due to the underlying block integrity structure, clients can initiate and verify an unlimited number of challenges.

In this paper, we propose Outsourced Proofs of Retrievability (OPOR) which enables an auditor to execute the POR protocol on behalf of the data owner and guarantees the security of data.Auditorsare liable to monitor the availability of theirfiles, users can verify the auditor's work at any point in time; we show that this verification can be much less frequent, and is more efficient when compared to theverification of existing POR.It correctly verify the availability of outsourced data and issues security SLA for the cloud users.

It is believed that supporting dynamic data operation is of great importance to the practical application of storage-outsourcing services. Using public verifiability and dynamic data operation support for cloud data storage, here a framework and an efficient construction for seamless integration of these two components in protocol design has been presented. Most of the existing works adopt weaker security models which usually do not consider reset attack, which are triggered by cloud storage server in the upload phase to violate the soundness of the scheme.Currently no existing scheme simultaneously provides provable security and enjoys desirable efficiency which means no scheme can resist reset attacks while supporting public verifiability and dynamic data operation simultaneously.

OPoR,a new PoR scheme with two Independent cloud servers. One server is for auditing and the other for data storage. The user is relieved from the computation of tax for files, i.e. outsourced to the cloud audit server. The files which are remotely stored in the cloud storage server or audited by the audit server.

In the proposed model security against reset attacks has been enhanced. It is the first PoR model that considers reset attacks for cloud storage.Remote data integrity in the cloud storage has been ensured using an efficient verification scheme. This scheme is proved to support dynamic data operation and public verifiability simultaneously.

II. PROOF OF RETRIEVABILITY MODEL

This model adopts spot -checking and error-correcting codes to ensure both Possession and Retrievability of data files in archive service systems.Public verifiability is not supported in this scheme which forces data owner to make many computational efforts to generate tags for the outsourced data. An improved PoR scheme has been designed with public verifiability based on BLS signature. This scheme used publicly verifiable homomorphic authenticators which are built from BLS signatures.

The PDP model has been extended to support provable updates to stored data files using rank based authenticated skip list and this scheme is fully dynamic version of the PDP solution

An efficient remote data verification scheme for supporting public verifiability and fully dynamic data operations simultaneously for PoR systems is our proposed solution. Unlike the previous works users need not compute the tags for the data outsourced thereby reducing computational overheads at user side .A detailed security analysis and efficiency analysis has been presented in this scheme.

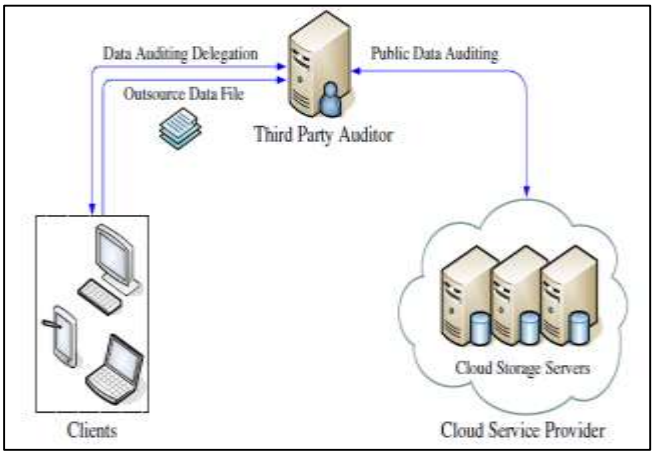

Fig. 1: Cloud data storage architecture

III. POR- FRAMEWORK

In a POR protocol, the client encodes the file before storing and enables bandwidth efficient challenge-response protocols to guarantee the data integrity. It usesa technique of "spot-checking" in the challenge-response protocol to detect adversarial behavior. A subset of file blocks is sampled and the results are returned to the client during each challenge. Thereturned results are checked using some additional information embedded into the file at encoding time.

POR framework assumes that the adversary replies correctly to a fraction of challenges 1 ¡ ² > 0 chosen uniformly at random from the challenge space.To be resilient to small adversarial corruption, an errorcorrectingcode (called outer code) is used to encode the full file.

If the client fails to download a file normally, it resorts to a POR-based extraction where the clients challenges the server and reconstructs the file from the values given by the server during its response. The concatenation of inner code and outer code is a classical technique for creating error-correcting codes with certain properties.The inner code may vary dynamically based according to the challenge format, the outer code serves in the initial file encoding, and is thus static over the lifetime of a file. The inner code imposes more computational burden to server than the outer code.

PORs can be classified into two main types based on the number of verifications they support over the lifetime of the system.

1) PORs that enable unlimited number of verifications, such as the SW scheme, are usually constructedby storing additional integrity values for file blocks. SW employs homomorphic integrity valuesthat can be aggregated over multiple blocks, resulting in a bandwidth-efficientPOR. The inner code can simply be an erasure code and the scheme can tolerate error rates non-negligibly close to 1. It is still essential to bound the adversarial corruption rate using a Phase-I protocol, since this ensures that a file can be extracted efficiently.
2) PORs that can verify a limited number of queries, such as the JK scheme, usually pre-compute theresponses to a set of challenges and embed them into the file encoding. The verification capability of these PORs is limited by the number of pre computed challenges embedded into the fileencoding. These types of POR cannot feasibly check all the responses received during Phase II, and thus need to employ error-correcting inner codes. In such protocols, we require an upper bound on the adversarial rate less than 1.The actually tolerable ² depends on the inner and outer code parameters. For such protocols to fit our framework, we need to assume that the adversary's corruption rate in Phase I is also limited by the same ² determined by the extraction capability in Phase II.

IV. OPOR: OUTSOURCED PROOFS OF RETRIEVABILITY

OPoR consists of auser U, and the data owner, who moves its data to a service provider and anew entity called the auditor which runs PoR with service provider on behalf of the user. It consists of five protocols Setup, Store, POR, CheckLog, and ProveLog.The difference is that the POR protocol not only outputs a decision on whether the POR has been correct, but also a log file. CheckLogprocedure allows the user to check if the auditor did his job during the runtime of the OPORScheme. ProveLog procedure allows the auditor to prove that if some problems occur.

The Setup Protocol: It generates for each of the different parties a public-private key pair. For each of the subsequent protocols and proceduresthat an involved party always uses as inputs its own secret key and the public keys of the other parties.

The Store Protocol: It takes the secret keys of the
Parties and a file M from the user to be stored. Theoutput needs to contain information that enables the execution of a POR protocol between auditor and the service provider on the one hand and enables the validation of the log files created by auditor on the other hand.

The POR Protocol: The auditorA and service providerS run a PoR protocol to convince the auditor that fileM is still retrievable from the service providerS.The input of A is the tag _A given by Store, and the input of the provider S is the stored copy of the file M.It holds that:
POR: [A : _A; S : M] → [A: ^,decA]
The protocol run is accepted by the auditor if decA = TRUE.

The CheckLog Algorithm: It enables the user to audit the auditor. It is a deterministic algorithm which takes verification key and log files as input and gives binary values as output, indicating if the log file is correct or not
dec^ := CheckLog(_U; ^).

The ProveLog Algorithm: It is a deterministic algorithm that complements the CheckLog procedure to ensure the correctness of the auditor in case of conflicts. It proves or disproves the honesty of auditor as it has access to the secret information of auditor.
dec^corr := ProveLog(_A; ^).

Correctness:The definition of correctness requires that if all parties are honest, then the auditor always, i.e., with probability 1, accepts at the end of each POR protocol run and likewise the user at the end of each CheckLog protocol run. This should hold for any choice of key pairs and for any file M €{0,1}.

Security Model: Security is formalized using the notion of an extractor algorithm that is able to extract the file in interaction with the adversary. The prover convinces the verifier with a sufficient level of probability then the file is actually stored. OPoR should provide security to the auditor also unlike in traditional PoR schemes.

V. ARCHITECTURE AND TECHNIQUES

The architecture consists of four entities: Data Owner(DO) ,who has large amount of data to be stored in the cloud; CSP, who provides data storage service and has enough storage space and computation resources; TPA,who has capabilities to manage or monitor the outsourced data under the delegation of DO; and authorized applications(AAs), who have the right to access and manipulate the stored data. Finally, application users can enjoy various cloud application services via these AAs.

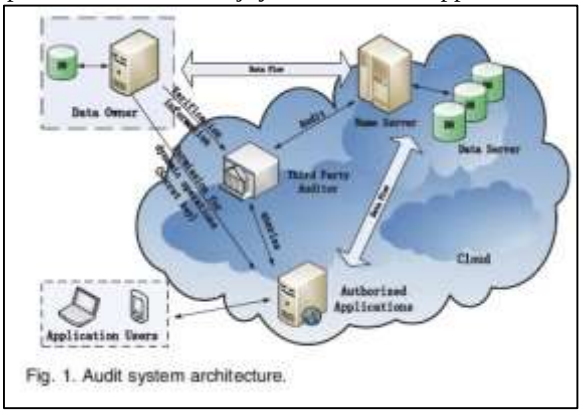

Fig. 2:Audit System architecture

TPA is reliable and independent through the following audit functions: TPA should be able to make regular checks on the integrity and availability of the delegated data at appropriate intervals; TPA should be able to organize, manage, and maintain the outsourced data instead of DOs, and support dynamic data operations for AAs; and TPA should be able to take the evidences for disputes about the inconsistency of data in terms of authentic records for all data operations.
Audit service is comprised of three processes:

– Tag generation. The client (DO) uses a secret key sk to preprocess a file, which consists of a collection of n blocks, generates a set of public verification parameters (PVPs) and IHT that are stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy.

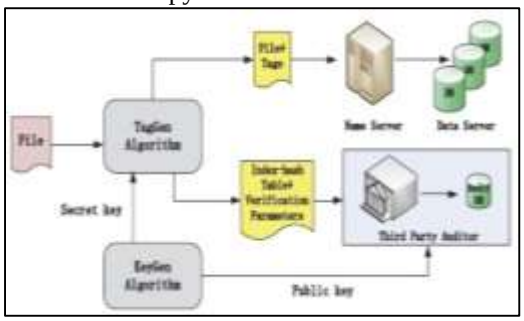

Fig. 3:Tag generation for outsourcing

– Periodic sampling audit. By using an interactive proof protocol of retrievability, TPA (or other applications) issues a "random sampling" challenge to audit the integrity and availability of the outsourced data in terms of verification information (involving PVP and IHT) stored in TPA.

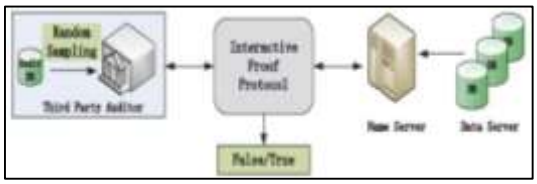

Fig. 2:Periodic sampling audit

– Audit for dynamic operations. An AA, who holds aDO's secret key sk, can manipulate the outsourced data and update the associated IHT stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the AAs and forge the valid audit records.

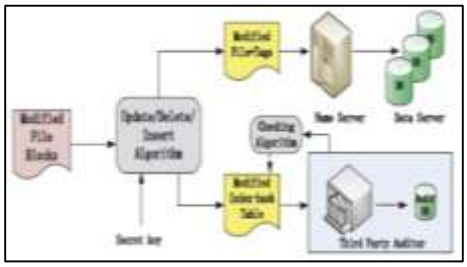

Fig. 2: Dynamic data operations and audit

The AAs should be cloud application servicesinside clouds for various application purposes, but theymust be specifically authorized by DOs for manipulatingoutsourced data and must present authentication information for TPA, any unauthorized modifications for data will bedetected in audit processes or verification processes.

The audit infrastructure enhances the credibility of CSSs, but not to increase DO's burden. Therefore, TPA should be constructed in clouds and maintained by a CSP. TPA must be secure enough to resist malicious attacks controlled to prevent unauthorized accesses even for internal members in clouds. TPA in clouds should be mandated bya trusted third party (TTP) Which improves the performance of an audit service and provides the DO with a maximum access transparency. This includes some procedures:KeyGen, TagGen,Update, Delete, Insert algorithms, as wellas an Interactive Proof Protocol of Retrievability.

A. Fragment Structure and Secure Tags

This technique is used to improve the storage efficiency and audit performance. An outsourced file F is split into n blocks and each block into 8 sectors with a signature tag. These tags and corresponding data can be used to construct a response in terms of the TPA's challenges in the verification protocol, such that this response can be verified without raw data. Secure tag is unforgeable by anyone except the original owner. Block-tag pairs are stored in CSP and theencrypted secrets are in TTP

B. Periodic Sampling Audit

This technique realizes the anomaly detection in a timely manner, as well as to rationally allocate resources. Given a randomly chosen challenge with a subset of the block indices and a random coefficient, an efficient algorithm is used to produce a constant-size response. This algorithm relies on homomorphic properties to aggregate data and tags into a constant-size response, which minimizes network communication costs. The audit activities are efficiently scheduled in an audit period, and a TPA merely needs to access small portions of files to perform audit in each activity.

C. Index-Hash Table

This technique supports dynamic data operations. It provides a higher assurance to monitor the behaviour of an untrusted CSP. It consists of serial number, block number, version number, and random integer and assure all records in the IHT differ from one anotherto prevent the forgery of data blocks and tags. Each record in the table is used to generate a unique hash value which in turn is used to create a secret key. The relationship between each block and the signature key must be cryptographically secure so that it is used to create the verification protocol.

VI. ALGORITHMS FOR AUDIT SYSTEM

- KeyGen takes a security parameter k as an input, and returns a public/secret key pair (pk,sk);
- TagGen (sk; F) takes a secret key sk and a file F, and returns a triple.

Whenever a data owner, DO wants to store a file in a storage server, and maintains a corresponding authenticated index structure at a TPA,using KeyGen(), the owner first generates public/secret keypair (pk,sk) by himself or the system manager, and sends his public key pk to TPA. The owner chooses a random secret key sk and invokes TagGen() to produce public verification information and signature tags. Finally, the owner sends verification scheme to TPA and CSP, respectively.

VII. PERFORMANCE

The implemenatation is done in Java 1.6 where Java Virtual Machine has 1GB of memory available for processing. The average latency time for the hard drive is 4.2ms and the average seek time is 10ms. We use the BSAFE library in Java for implementing the cryptographic operations. The encoding algorithm achieves a throughput of around 3MB/s, and we observe that encoding time grows linearly with file size. The outer error-correcting layer is responsible for most of the encoding overhead. The outer code encoding time can be reduced by reducing the outer code distance, at the expense of checking more challenges in Phase I of our framework.

VIII. CONCLUSION

In our proposed scheme, OPoR a trustworthy audit server is used to preprocess and upload on behalf of the users. Thus reduces the computation overhead significantly. Data integrity verification and updating the outsourced data is done by the client audit server. This approach enhances the security against reset attacks in the upload phase along with public verifiability and dynamic data operations. Furthermore the proposed scheme can be enhanced to reduce the trust on the cloud audit server for more generic applications, security against reset attacks can be strengthened and efficient constructions for less storage and communication cost can be found.This paper put forward a scheme for data storage security verification based on third party in Cloud Computing to meet demand of user in data integrity, data confidentiality, data recovery and retrieval, credibility control of TPA. As a future work,it is worth exploring further optimizations to improve the encoding throughput.

REFERENCES

[1] A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In Proc. ACM CCS
[2] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. ACM Conf. Computer and Communications Security (CCS '07), pp. 584-597, 2007.
[3] H. Shacham and B. Waters, "Compact proofs of retrievability,"in ASIACRYPT '08:Proceedings of the 14th InternationalConference on the Theory and Application of Cryptology andInformation 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] Q. Zheng and S. Xu, "Secure and efficient proof of storage withDeduplication ," in CODASPY, 2012, pp. 1–12.
[6] Cloud Computing: Cloud Security Concerns.http://technet.microsoft.com/en-us/ magazine/hh536219.aspx.
[7] CASH, D., KÜPÇÜ, A., AND WICHS, D. Dynamic Proofs ofRetrievability via Oblivious RAM. In EUROCRYPT (2013),pp. 279–295.
[8] BOWERS, K. D., JUELS, A., AND OPREA, A. Proofs of retrievability: theory and implementation. In CCSW (2009),pp. 43–54.
[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,"Proc. IEEE INFOCOM, pp. 1-9, 2010.
[10] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity Auditing of Outsourced Data," Proc. 33rd Int'l Conf. Very Large Databases (VLDB), pp. 782-793, 2007.