IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 6, June 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

Data Integrity and Security of Storage Management in Cloud Computing

Dr. Shubhangi D.C[1], Sham [2]

[1]H.O.D, Department of Computer Science and Engineering, VTU Regional Centre, Kalaburagi, Karnataka, INDIA.
[2]P.G.Student, Department of Computer Science and Engineering, VTU Regional Centre, Kalaburagi, Karnataka, INDIA.

Abstract

Cloud Computing moves the application software and databases to the centralized immensely colossal data centers, where the management of the data and accommodations may not be plenarily trustworthy. In this work, we study the quandary of ascertaining the integrity of data storage in Cloud Computing. To reduce the computational cost at utilizer 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 immensely colossal for the users with resource-constrained contrivances to compute the public authentication tags of file blocks. To tackle the challenge, we propose OPoR, an incipient cloud storage scheme involving a cloud storage server and a cloud audit server, where the latter is surmised to be semi-veracious. In particular, we consider the task of sanctioning the cloud audit server, on behalf of the cloud users, to pre-process the data after uploading to the cloud storage server and later verifying the data integrity. It outsources the cumbersomely hefty computation of the tag generation to the cloud audit server and eliminates the involution of utilizer in the auditing and in the preprocessing phases. Furthermore, we reinforce the Proof of Retrievabiliy (PoR) model to fortify dynamic data operations, as well as ascertain security against reset attacks launched by the cloud storage server in the upload phase.

Keywords: cloud computing, computer centres, data integrity, security of data, storage management.

1. 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, and usage based pricing. In particular, the ever more frugal and more potent processors, together with the “software as a service” (SaaS) computing architecture, are transforming data centers into pools of computing accommodation on an immensely colossal scale.

Albeit having appealing advantages as a promising accommodation platform for the Internet, this incipient 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 most astronomically immense concerns with remote data storage is that of data integrity verification at untrusted servers. For instance, the storage accommodation provider may decide to obnubilating such data loss incidents as the Byzantine failure from the clients to maintain a reputation. What is more solemn is that for preserving maxima and storage space the accommodation provider might deliberately discard infrequently accessed data files which belong to a mundane client. Considering the astronomically immense size of the outsourced electronic data and the client's constrained resource capability, the core of the quandary can be generalized as how can the client find an efficient way to perform periodical integrity verification without the local replica of data files. In order to surmount this quandary, many schemes have been proposed under different system and security models. In all these works, great efforts have been made to design solutions that meet requisites: high scheme efficiency, stateless verification, unbounded utilization of queries and retrievability of data, etc. According to the role of the verifier in the model, all the schemes available fall into two categories: private verifiability and public verifiability. Albeit achieving higher efficiency, schemes with private verifiability impose computational burden on clients. On the other hand, public verifiability alleviates clients from performing an abundance of computation for ascertaining the integrity of data storage. To be categorical, clients are able to delegate a third party to perform the verification without devotion of their computation resources. In the cloud, the clients may crash unexpectedly or cannot afford the overload of frequent integrity checks. Thus, it seems more rational and practical to equip the verification protocol with public verifiability, which is expected to play a more paramount role in achieving better efficiency for Cloud Computing.

2. Related Work

Recently, much research effort has been devoted largely to ensure the security of cloud computing. In particular, we consider the task of sanctioning the cloud audit server, on behalf of the cloud users, to pre-process the data afore uploading to the cloud storage server and later verifying the data integrity. OPoR outsources the heftily ponderous computation of the tag generation to the cloud audit server and eliminates the involution of utilizer in the auditing and in the preprocessing phases. Furthermore, we reinforce the Proof of Retrievably (PoR) model to fortify dynamic data operations, as well as ascertain security against reset attacks launched by the cloud storage server in the upload phase.

PDP model[1] was introduced which allows the client to verify the data stored at a untrusted servers without reacquiring it. This model acquires probabilistic proofs of possession by examining random sets of blocks from the server, which vitally reduce the input/output costs. The client keeps up a stabile amount of metadata to verify the proof. The challenge/response compact transmits a small, constant amount of data, which decreases network communication. Thus, the PDP design for remote data checking backs abundant data sets in widespread storage systems. The two provably-secure PDP schemes that are more effective than former solutions, even when related with schemes that attain weaker guarantees. In distinct, the burden at the server is less (or even constant), as opposed to linear in the size of the data. Experiments implementation of PDP reveal that the performance of PDP is confined by disk input/output, not by cryptographic computation.

POR[2] is a concise proof by a prover(server) or file system to a client (verifier) that a file F stored at server is intact, in the belief that the client can fully recapture it. As PORs induce lower communication multiplicity than transferring a file F, present at the attractive building block for high-security of remote storage systems. Here we come up with a theoretical structure for the design of PORs. THE present design upgrades the previously recommended POR architecture of Juels-Kaliski and Shacham-Waters, and also yields content on the limitations of previous theoretical models for PORs. It supports thourghly Byzantine adversative model, convening only the limits which are fundamentals of POR's, that the attacker's error rate be restricted when the client seeks to extract F.The tactics to support efficient protocols across the achievable range of , up to range close to 1. Here we introduce a new alternative on the Juels-Kaliski protocol and determine a prototype implementation and demonstrate a practical encoding for files F whose size exceeds that of client main memory. full security for POR against the attackers[3] was introduced by shortest query and response time of POR using public verifiability and shortest response with private verifiability.Using MAC security level is enhanced .It is sufficient for the client to retrieve retrieves a few blocks together with their MACs and check, using his secret key, that these blocks are correct. K. D. Bowers[4],in this POR model a prover can verify that the file is intact and the client can recover the whole data .The client can retrieve all of F from the server with high probability and a technique called "spot-checking" to check error rate of a large files was introduced.M. Naor[5],here the problem of storing a file at the remote server, to know that file has been corrupted a end user stores a constant amount of metadata. The user must store this data in such way that it should allow him to verify the data without reading a entire file the same (tight) lower bound applies also to that problem.E.C. Chang[6],a client or a verifier having a small amount of storage space check s periodically that the remote server keeping a file safely,but an untrusted server may discard the request.So remote integrity check(RIC) with a combination of RSA model.In this POR scheme there is a time extractor can obtain the data by multiple verifications using error corrected code.MA Shah,the present generation customers uses the online services of google,amazon etc for the storage of their valuable data.The customer must entirely trust the service provider that it maintains the integrity of data.A service provider may hide the data loss incidents so in order to overcome this a protocol called TPA was introduced to verify the data at remote storage and the original data is never made available to TPA.So the TPA avoid the burden of frequent integrity checks at the user side. Y. Zhu[8].introduced dynamic audit server with integrity verification at the untrusted server. The audit server supports dynamic data operations such as insert, delete and update and also with the help of indexed hash table file corruption is detected this leads to the lower computational cost and requires an additional storage for integrity verification.B wang[9],addressed the problem of integrity of a shared data because with the help of cloud service it is common that the data is shared among multiple users,the public auditing of this data is a major concern to maintain a identity privacy. Ring signature was introduced to compute the information which must be verified needed for the integrity of data and the identity of each block is kept private from TPA.[10], with the help of TPA the burden caused on client to verify the data was reduced and also the auditing task of client was eliminated. The previous architectures of POR does not support integrity check and dynamic update concurrently but the present model solves this problem with the help of MHT for tag block authorization

System Architecture:

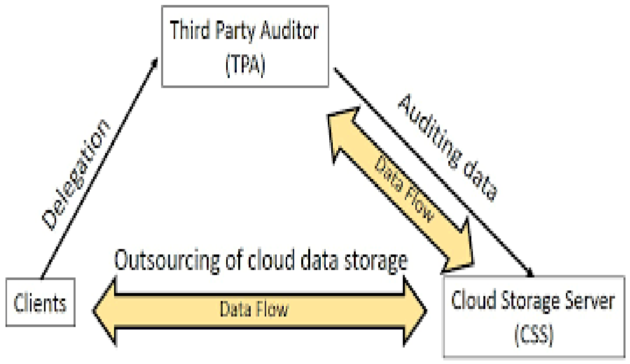

Third Party Auditor (TPA)
Delegation
Auditing data
Data Flow
Outsourcing of cloud data storage
Clients
Data Flow
Cloud Storage Server (CSS)

Fig 1: System Architecture

Client: An entity that 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 client's data. The CSS is required to provide integrity proof to the clients or cloud audit server during the integrity checking phase.

Cloud Audit Server (CAS):A TPA, 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. In this system, the cloud audit server also generates all the tags of the files for the users before uploading to the cloud storage server.

3. Methodology

In our system both public verifiability and fully dynamic data operation are supported.

//////////The predefined parameters in construction are P-public key,S-secret key,J-input parameter ,TF-tags of file,K-prover

(P, S) ← **Setup**(1,J). It takes as input security parameter 1,k,returns public parameters and the key pair of the cloud audit server.

(F ∗, TF) ← **Upload**(S, F). There are two phases in this algorithm. In the first phase, the client uploads TF data file F to the cloud audit server, where F is an ordered collection of blocks {Mi}. In the second phase, the file F is re-uploaded to the cloud storage server by the cloud audit server: it takes as input the private key sk and F, and outputs the signature set Φ, which is an ordered collection of signatures {σi} on {Mi}. We denote the stored file F ∗ = {F, Φ}. It also outputs metadata-the root R of a Merkle hash tree from {Mi} and the signature t = sigsk(h(R)) as the tag of F ∗ . Notice that the storage server stores (F∗, TF), but the audit server (the client) only keeps t as receipt.

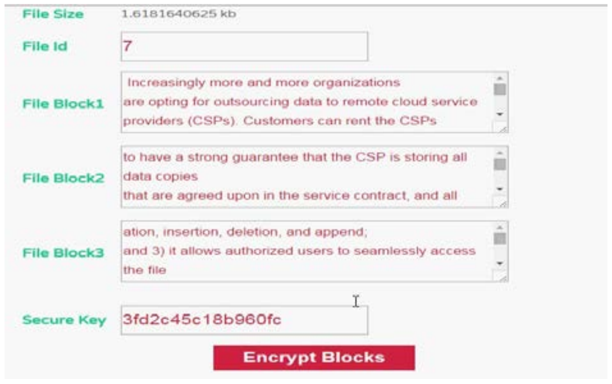
1/0 ← **Integrity Verification**{K(P, F∗ , TF) V (P, TF)}.This is an interactive protocol for integrity verification of a file F ∗ with tag TF. The cloud storage server plays the role of prover K with input the public key P, a stored file F and a file tag TF. The cloud audit server plays the role of verifier V with input P and TF. At the end of the protocol, V outputs TRUE (1) if F ∗ passes the integrity verification, or F ALSE (0) otherwise.

(F∗, TF) ← **Update** {K(P, Fˆ∗ ,TFˆ) K (S,TF, update ˆ)}. This is an interactive protocol for dynamic update of a file Fˆ∗ with tag TFˆ. The cloud storage server plays the role of prover K with input the public key P, a stored file Fˆ∗ , and a file tag TF. The cloud audit server plays the role of verifier V with input the private key S, TF, and an data operation request "update" from the client. At the end of the protocol, V outputs a file tag TF of the updated file F ∗ if K gives a valid proof for the update, or F ALSE (0) otherwise.

- **Correctness.** A PoR scheme is correct if the following two conditions hold:
  If (F∗,TF) ← Upload(S,F) then IntegrityVerify{K(P, F∗ , TF) V (P, TF)} = 1.

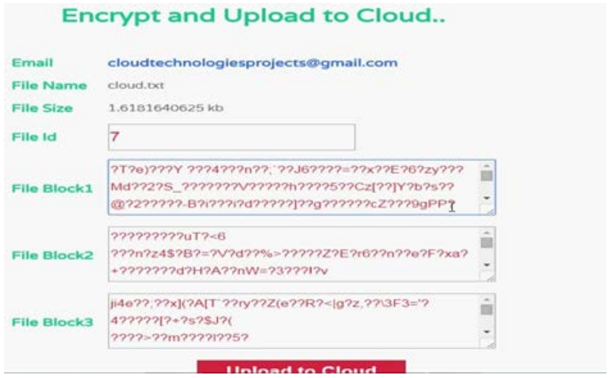
Since the cloud audit server is fully trusted in the two-server architecture, we allow it to generate the key pairs on behalf of the clients in the setup phase. However, it might be undesirable to place full trust on the cloud audit server in some outsourcing tasks. Consider the following scenario: one storage service is available to the clients on a pay-per-use basis, and the audit server may upload a file, intentionally or mistakenly, on behalf of one client who did not ask for storing that file. One solution for such applications is utilizing a proxy signature scheme supporting delegation by warrant to delegate the signing right of the clients to the cloud audit server for each usage. The warrant to the audit server can be the hashed value of the uploaded file as a credential of the delegation

.4. Result and Discussion



File Size	1.6181640625 kb
File Id	7
File Block1	Increasingly more and more organizations are opting for outsourcing data to remote cloud service providers (CSPs). Customers can rent the CSPs
File Block2	to have a strong guarantee that the CSP is storing all data copies that are agreed upon in the service contract, and all
File Block3	ation, insertion, deletion, and append; and 3) it allows authorized users to seamlessly access the file
Secure Key	3fd2c45c18b960fc

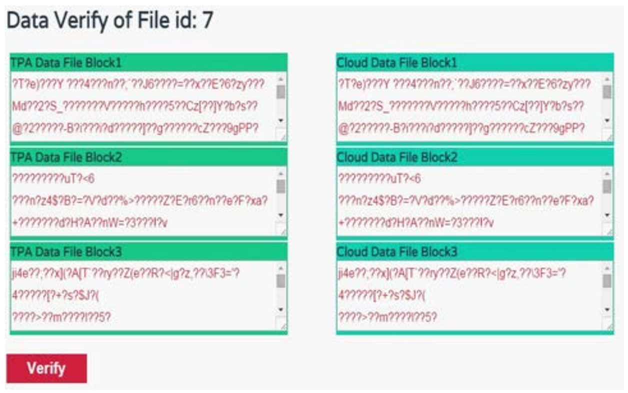
Fig 2: File With Secrete Key.



Encrypt and Upload to Cloud..

Email	cloudtechnologiesprojects@gmail.com
File Name	cloud.txt
File Size	1.6181640625 kb
File Id	7
File Block1	?T?e)???Y ???4???n??;`??J6????=??x??E?6?zy??? Md???2?S_???????V?????h????5??Cz[??]Y?b?s?? @?2?????-B?i???i?d?????]??g??????cZ???9gPP?
File Block2	?????????uT?<6 ???n?z4$?B?=?V?d??%>?????Z?E?r6??n??e?F?xa? +???????d?H?A??nW=?3???I?v
File Block3	ji4e??;??x](?A[T`??ry??Z(e??R?<|g?z,??\3F3='? 4?????[?+?s?$J?(????>??m????I??5?

Fig 3: File with Encrypted Format before Uploading.



Data Verify of File id: 7

TPA Data File Block1

```
?T?e)???Y ???4???n??,`??J6????=??x??E?6?zy???
Md??2?S_??????V?????h????5??Cz[??]Y?b?s??
@?2?????-B?i???i?d?????]??g??????cZ???9gPP?
```

Cloud Data File Block1

```
?T?e)???Y ???4???n??,`??J6????=??x??E?6?zy???
Md??2?S_??????V?????h????5??Cz[??]Y?b?s??
@?2?????-B?i???i?d?????]??g??????cZ???9gPP?
```

TPA Data File Block2

```
????????uT?<6
???n?z4$?B?=?V?d??%>?????Z?E?r6??n??e?F?xa?
+???????d?H?A??nW=?3???I?v
```

Cloud Data File Block2

```
????????uT?<6
???n?z4$?B?=?V?d??%>?????Z?E?r6??n??e?F?xa?
+???????d?H?A??nW=?3???I?v
```

TPA Data File Block3

```
ji4e??,??x](?A[T`??ry??Z(e??R?<|g?z,??\3F3='?
4?????[?+?s?$J?(
????>??m????I??5?
```

Cloud Data File Block3

```
ji4e??,??x](?A[T`??ry??Z(e??R?<|g?z,??\3F3='?
4?????[?+?s?$J?(
????>??m????I??5?
```

Fig 4: TPA Data Verifying Page

In the first experiment, the computational overhead for the tag generation of files at the cloud audit server is evaluated. We have not checked the computational overhead at users because it only needs the computation of a digital signature, which is very small compared with the computation of the tags. The reason is that the most overhead computation has been delivered to the cloud audit server. Three different numbers of s are chosen in the experiment to show the effect on the efficiency of the time cost.From Fig. 5, the time cost grows when the number of s decreases. The average time cost for file with size 50KB is 5s. Compared with the previous related work the computational overhead at users is outsourced to the cloud audit server.

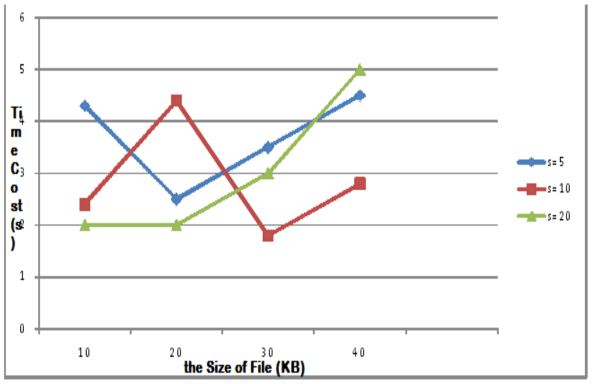


Fig 5: Graphical representation of Tag generation time.

5. Conclusion

This paper proposes ,a new proof of retrievability for cloud storage, in which a trustworthy audit server is introduced to preprocess 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 verifiability and dynamic data operation simultaneously.

6. 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 retrievability," 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] Q. Wang, K. Ren, S. Yu, and W. Lou, "Dependable and secure sensor data storage with dynamic integrity assurance," ACM Transactions on Sensor Networks, vol. 8, no. 1, pp. 9:1–9:24, Aug. 2011. [Online]. Available: http: //doi.acm.org/10.1145/1993042.1993051
[11] 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.
[12] 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.