ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

A Data Integrity Proofs in Cloud Storage with High Performance

[1] K.Sai Gowtham , [2]S.Suresh Babu
[1]Final M Tech Student, [2]Asst Professor
[1,2] Dept of Computer Science and Engineering
[1,2] Sri Mittapalli College of Engineering–Tummalapalem,Guntur,Guntur(dt),
Andhra Pradesh, India.

Abstract:
Cloud computing has been envisioned as the de-facto solution to the rising storage costs of IT Enterprises. With the high costs of data storage devices as well as the rapid rate at which data is being generated it proves costly for enterprises or individual users to frequently update their hardware. Apart from reduction in storage costs data outsourcing to the cloud also helps in reducing the maintenance. Cloud storage moves the user's data to large data centers, which are remotely located, on which user does not have any control. However, this unique feature of the cloud poses many new security challenges which need to be clearly understood and resolved. One of the important concerns that need to be addressed is to assure the customer of the integrity i.e. correctness of his data in the cloud. As the data is physically not accessible to the user the cloud should provide a way for the user to check if the integrity of his data is maintained or is compromised. In this paper we provide a scheme which gives a proof of data integrity in the cloud which the customer can employ to check the correctness of his data in the cloud. This proof can be agreed upon by both the cloud and the customer and can be incorporated in the Service level agreement (SLA). This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients. And also we propose the network bandwidth is also minimized as the size of the proof is comparatively very less (k+1 bit for one proof). It should be noted that our scheme applies only to static storage of data. It cannot handle to case when the data need to be dynamically changed. Hence developing on this will be a future challenge. Also the number of queries that can be asked by the client is fixed apriori. But this number is quite large and can be sufficient if the period of data storage is short. It will be a challenge to increase the number of queries using this scheme.

Keywords-SLA, Cloud Computing, IT Enterprises, Static storage.

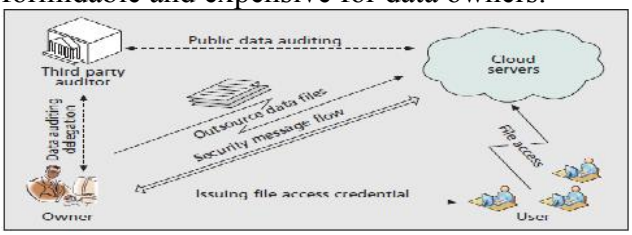
1. Introduction:

Cloud computing has been envisioned as the next generation architecture of the IT enterprise due to its long list of unprecedented advantages in IT: on demand self-service, ubiquitous network access, location-independent resource pooling, rapid resource elasticity, usage-based pricing, and transference of risk [1]. One fundamental aspect of this new computing model is that data is being Centralized or outsourced into the cloud. From the data owners' perspective, including both individuals and IT enterprises, storing data remotely in a cloud in a flexible on-demand manner brings appealing benefits: relief of the burden of storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, personnel maintenance, and so on [2]. While cloud computing makes these advantages more appealing than ever, it also brings new and

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

challenging security threats to the outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing actually relinquishes the owner's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they still face a broad range of both internal and external threats to data integrity. Outages and security breaches of noteworthy cloud services appear from time to time. Amazon S3's recent downtime [3], Gmail's mass email deletion incident [4], and Apple Mobile Me's post-launch down time [1] a r e all s u c h examples. Second, for benefits of their own, there are various motivations for CSPs to behave unfaithfully toward cloud customers regarding the status of their outsourced data. Examples include CSPs, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed [2], or even hiding data loss incidents to maintain a reputation [4]. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large scale data storage .it does not offer any guarantee on data integrity and availability . This problem, if not properly addressed, may impede successful deployment of the cloud architecture.

As data owners no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted [3]. In particular, simply downloading the data for its integrity verification is not a practical solution due to the high cost of input/output (I/O) and transmission across the network. Besides, it is often insufficient to detect data corruption only when accessing the data, as it does not give correctness assurance for un accessed data and might be too late to recover the data loss or damage. Considering the large size of the outsourced data and the owner's constrained resource capability, the tasks of auditing the data correctness in a cloud environment can be formidable and expensive for data owners.



Able to just use cloud storage as if it is local, without worrying about the need to verify its integrity. Hence, to fully ensure data security and save data owners' computation resources, we propose to enable publicly auditable cloud storage services, where data owners can resort to an external third party auditor (TPA) to verify the outsourced data when needed. Third party auditing provides a transparent yet cost-effective method for establishing trust between data owner and cloud server. In fact, based on the audit result from a TPA, the released audit report would not only help owners to evaluate the risk of their subscribed cloud data services, but also be beneficial for the cloud service provider to improve their cloud based service platform . In a word, enabling public risk auditing protocols will play an important role for this nascent cloud economy to become fully established; where data owners will need ways to assess risk and gain trust in the cloud.

Recently, great interest has been shown in ensuring remotely stored data integrity under different system and Security models. Some of the work has already been promoting the development of public audit ability for existing cloud data storage services. However, it is not feasible yet. On one hand, data owners are currently not sophisticated enough to demand risk assessment; on the other hand, current commercial cloud vendors do not provide such a

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

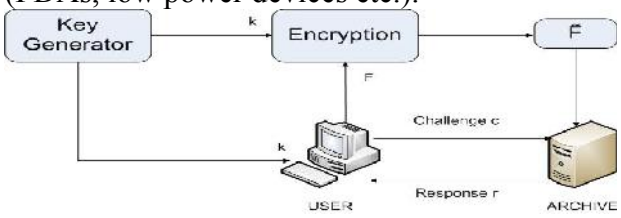
third party auditing interface to support a public auditing service. This article is intended as a call for action, aiming to motivate further research on dependable cloud storage services and enable public auditing services to become a reality. We start by suggesting a set of systematically and cryptographically desirable properties that should apply to practical deployment for securing the cloud storage on behalf of data owners. We sketch a set of building blocks, including recently developed cryptographic primitives (e.g. homomorphism authenticator), to ensure these strong security properties, which could form the basis of a publicly auditable secure cloud data storage system.

2. Existing System

The simplest Proof of derivability (POR) scheme can be made using a keyed hash function hk(F). In this scheme the verifier, before archiving the data file F in the cloud storage, pre-computes the cryptographic hash of F using hk(F) and stores this hash as well as the secret key K. To check if the integrity of the file F is lost the verifier releases the secret key K to the cloud archive and asks it to compute and return the value of hk(F). By storing multiple hash values for different keys the verifier can check for the integrity of the file F for multiple times, each one being an independent proof. Though this scheme is very simple and easily implementable the main drawback of this scheme are the high resource costs it requires for the implementation. At the verifier side this involves storing as many keys as the number of checks it want to perform as well as the hash value of the data file F with each hash key. Also computing hash value for even a moderately large data files can be computationally burdensome for some clients(PDAs, mobile phones, etc). As the archive side, each invocation of the protocol requires the archive to process the entire file F. This can be computationally burdensome for the archive even for a lightweight operation like hashing. Furthermore, it requires that each proof requires the proverb to read the entire file F - a significant overhead for an archive whose intended load is only an occasional read per file, were every file to be tested frequently [3]. Ari Juels and Burton S. Kaliski Jr proposed a scheme called Proof of irretrievability for large files using"sentinels"[3]. In this scheme, unlike in the key-hash approach scheme, only a single key can be used irrespective of the size of the file or the number of files whose irretrievability it wants to verify. Also the archive needs to access only a small portion of the file F unlike in the key-has scheme which required the archive to process the entire file F for each protocol verification. This small portion of the file F is in fact independent of the length of F. The schematic view of this approach is shown in Figure1. In this scheme special blocks (called sentinels) are hidden among other blocks in the data file F. In the setup phase, the verifier randomly embeds these sentinels among the data blocks. During the verification phase, to check the integrity of the data file F, the verifier challenges the proverb (cloud archive) by specifying the positions of a collection of sentinels and asking the proverb to return the associated sentinel values. If the proverb has modified or deleted a substantial portion of F, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. To make the sentinels indistinguishable from the data blocks, the whole modified file is encrypted and stored at the archive. The use of encryption here renders the sentinels indistinguishable from other file blocks. This scheme is best suited for storing encrypted files. As this scheme involves the encryption of the file F using a secret key it becomes

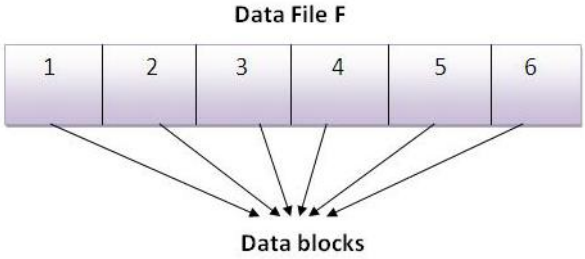
ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

computationally cumbersome especially when the data to be encrypted is large. Hence, this scheme proves disadvantages to small users with limited computational power (PDAs, mobile phones etc.). There will also be storage overhead at the server, partly due to the newly inserted sentinels and partly due to the error correcting codes that are inserted. Also the client needs to store all the sentinels with it, which may be storage overhead to thin clients (PDAs, low power devices etc.).



3. PROPOSEDWORK

We present a scheme which does not involve the encryption of the whole data. We encrypt only few bits of data per data block thus reducing the computational overhead on the clients. . A data file F with 6 data blocks The client storage overhead is also minimized as it does not store any data with it. Hence our scheme suits well for thin clients. In our data integrity protocol the verifier needs to store only a single cryptographic key - irrespective of the size of the data file F- and two functions which generate a random sequence. The verifier does not store any data with it. The verifier before storing the file at the archive, preprocesses the file and appends some meta data to the file and stores at the archive. At the time of verification the verifier uses this meta data to verify the integrity of the data. It is important to note that our proof of data integrity protocol just checks the integrity of data i.e. if the data has been illegally modified or deleted. It does not prevent the archive from modifying the data. In order to prevent such modifications or deletions other schemes like redundant storing etc, can be implemented which is not a scope of discussion in this paper.



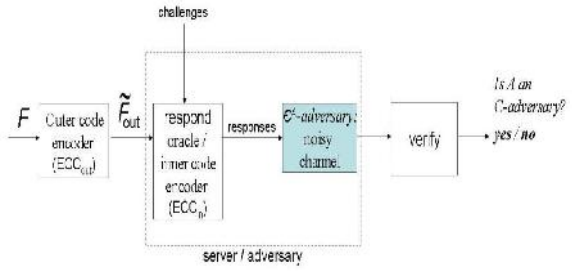
3.1 POR framework: Key ideas

It is useful to think of a POR in our framework as a two-phase process:
Phase I: Ensuring an ϱ-adversary
In the first phase of a POR (depicted in Figure 1), the client performs a series of challenge-response interactions with the server A over file F˜ out (i.e., the encoding of F under the outer code ECCout), with the aim of detecting the condition ϱ A > ϱ. To challenge the server, the client computes c = challenge (η; κ, ω)[π], sends c to the server, receives a response r, and then computes verify((c, r, η); κ, ω) to check the response of the adversary. The client repeats this process qc times, and rejects if any response is incorrect. Otherwise the client accepts. Assuming that challenge selects c ∈U C, the probability that an adversary A is accepted but is not an ϱ-adversary, i.e., ϱ
A> ϱ, is
< λ = (1 − ϱ)
Qc . The value λ can be made arbitrarily small, with an appropriately large qc. The JK protocol checks adversarial responses by precomputing a challenge set {ci} q i=1 ∈U C and storing verifying data—sentinels
or MACs—in the encoded file for download by the client. The Lilli bridge et al. and NR constructions check adversarial responses by verifying MACs on file blocks. Thus, both of these constructions also select c ∈U C. The SW scheme omits Phase I, i.e., implicitly assumes an ϱ-adversary. Remark. In practice, the server

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

may initially be honest, but turn bad at some point and be replaced by an adversary A. To deal with such a dynamic adversary, the client may spread out its challenges over time. For example, the client might initiate a challenge every day. If Phase I is tuned to achieve a particular λ for qc = 50, then,
the condition ϱ A > ϱ will be detected with probability at least 1−λ within the first 50 days after the server has turned adversarial.

Phase II: Extracting F from an ϱ-adversary Assuming an honest server, a client can simply download F— and verify its correctness via an appended MAC or digital signature. If this fails, the client can download the encoded file F˜ η and try to correct it using the outer error-correcting layer. Failing that, given an ϱ-adversary, it is possible for the client to retrieve F via extract, executing a series of challenges and decoding F from the responses. Note that in this phase, the client may not be able to verify the correctness of the responses it receives (in particular, for protocols with a bounded number of verifications): In this case, it relies on the ϱ-bound on A for successful decoding. In our general framework, there are two levels of error-correction: • The outer code: This is a (n, k, d2)-error-correcting code ECCout applied to F to compute F˜ out, the error-corrected portion of F output by encode. Usually, for large files, n < m, and to encode a file of size m we need to resort to a well-known technique, called striping: the file is divided into stripes of size k blocks each and each stripe is encoded under ECCout. In the rest of the paper, when we encode the file with the outer code, we implicitly mean that striping is performed if necessary. • The inner code: This (w, v, d1)-error-correcting code ECCin represents a second layer of error-correction in the challenge response interface for a POR. The function respond(s, u) applies ECCin to the set s of message blocks specified in a challenge; the value u ∈ W specifies which symbol of the corresponding codeword should be returned to the client. In this view, the adversary is a noisy channel with error probability at most ϱ. We may think of the adversary as intercepting transmissions from a (correct) oracle for respond to the client. When the client submits the i-th challenge c = (s, u), the respond oracle computes the correct response r. If ϱ I s,u = 1, then A corrupts the response in the channel; otherwise, the adversary leaves r unchanged.

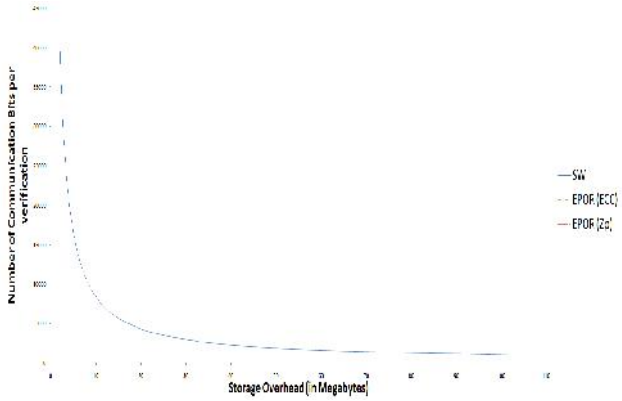


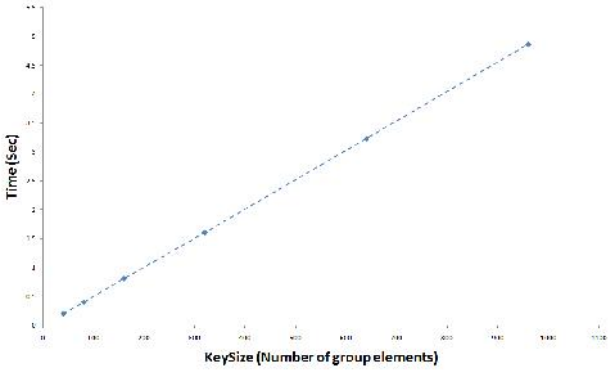
The goal of employing two levels of error correction in the design of our POR framework is to correct the adversarial error ϱ. The effect of the inner code is to drive down the adversarial error ϱ to some error value ϱ′ < ϱ. The outer code then corrects this residual error ϱ′ . Thus, the stronger the inner code, the weaker the outer code we need to employ. The outer code needs to be an adversarial
Error-correcting code (defined in the full version of the paper [4]). Intuitively, an adversarial code transforms an ϱ′ computationally bounded adversary to a random one, i.e., one in which the adversary has no better chance of corrupting codeword symbols than choosing at random.
Our PORs are designed to be effective against a maximum adversarial error ϱ. In the next section, we will show examples of protocols that tolerate ϱ values non-negligibly close to 1, as well as protocols that require an upper bound on ϱ less than 1 2 . There are two main types of POR protocols that we can construct in our framework:

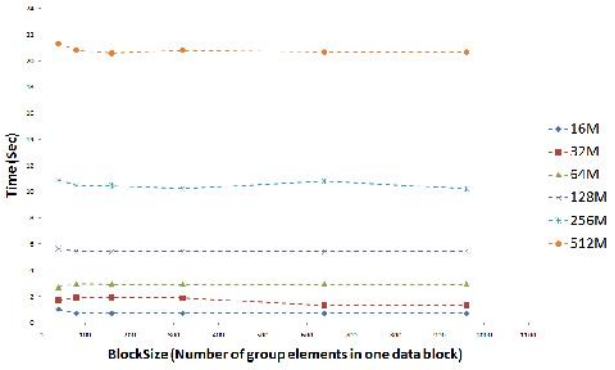
ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

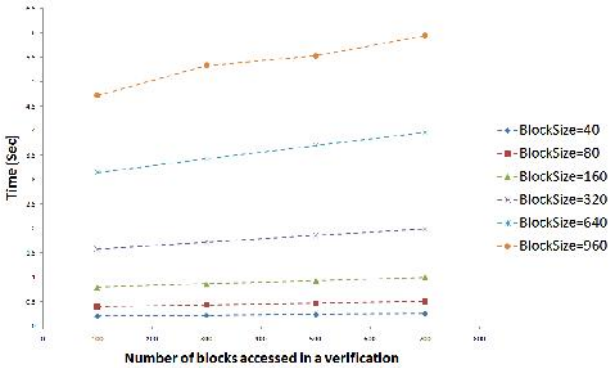
1. In protocols that enable an unbounded number of server verifications (e.g., SW), the responses to challenges in Phase II can be verified. In this case, the inner code can simply be an erasure code and can therefore tolerate error rates ϱ no negligibly close to 1. We describe in Section 4 a generalization of SW that fits into our framework. Even if, theoretically, such protocols do not need to employ Phase I to limit ϱ (since they can extract for ϱ close to 1), in practical settings obtaining a Phase-I bound on the adversarial corruption rate is still valuable to ensure efficient extraction.
2. In protocols in which a limited number of responses can be verified, we need to employ an error-correcting inner code in order to correct arbitrary server responses. In such protocols, we need to set an upper bound on the adversarial rate less than 1 2 and dependent on both the inner and outer code parameters. For such protocols to fit our framework, we need to assume that the adversary's corruption rate during extraction is also limited by the same ϱ < 1 2. Under this assumption, either the adversary corruption rate is greater than ϱ, and will be detected with high probability in Phase I, or ϱ is low enough so that we can extract F successfully in Phase II. We present a more efficient version of JK t hat employs a limited number of verifications assuming an upper bound on ϱ given by the error correction rate of the inner code. Despite such restrictions, we show its practical advantages in storage overhead and proof costs compared to SW for values of ϱ within the error-correction capabilities of the inner and outer codes. Now the full storage and successful extraction process for the client is as follows. We let the superscript denote a corrupted file

4. EXPERIMENTAL RESULTS

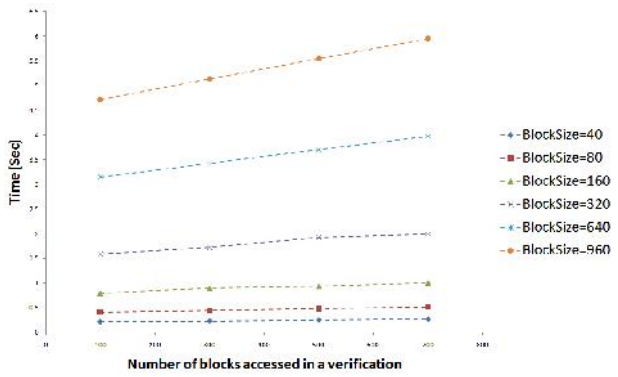








ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)



5. CONCLUSIONS AND FUTURE WORK

In this paper we have worked to facilitate the client in getting a proof of integrity of the data which he wishes to store in the cloud storage servers with bare minimum Costs and efforts. Our scheme was developed to reduce the computational and storage overhead of the client as well as to minimize the computational overhead of the cloud storage server. We also minimized the size of the proof of data integrity so as to reduce the network bandwidth consumption. At the client we only store two functions, the bit generator function g, and the function h which is used for encrypting the data. Hence the storage at the client is very much minimal compared to all other schemes [4] that were developed. Hence this scheme proves advantageous to thin clients like PDAs and mobile phones. The operation of encryption of data generally consumes a large computational power. In our scheme the encrypting process is very much limited to only a fraction of the whole data thereby saving on the computational time of the client. Many of the schemes proposed earlier require the archive to perform tasks that need a lot of computational power to generate the proof of data integrity [3]. But in our scheme the archive just need to fetch and send few bits of data to the client. And also evaluate the performance of the cloud storage performance.

REFERENCES

[1] E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and integrity in outsourced databases,” Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.
[2] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in SP ’00: Proceedings of the 2000 IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2000, p. 44.
[3] A. Juels and B. S. Kaliski, 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.
[4] 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.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Katuri Sai Gowtham received his B.Tech Degree In CSE From Sri Mittapalli College Of Engineering Tummalapalem Guntur (Dt), in 2010, the M.Tech. Degree in CSE from Sri Mittapalli College Of Engineering Tummalapalem, Guntur (Dt), at present, he is engaged in **"A Data Integrity Proofs In Cloud Storage With High Performance".**

Suresh Babu Siriparapu Pursuing Ph.D (CSE) in the area of Digital Image Processing at JNTUK, Kakinada. M.Tech (CSE) from J.B. Institute of Engineering and Technology, affiliated to JNTU Hyderabad in the year 2009. B.Tech (CSE) from Vignan's Engineering College affiliated to JNTU Hyderabad in the year 2005. Currently working as an Assistant Professor in CSE Dept. at Sri Mittapalli College of Engineering College, Thummalapalem, NH-5, Guntur(Dt.,) Worked as an Assistant Professor in CSE Dept. at Narasaraopeta Engineering College, Narasaraopet, from 10th June 2009 to 30th April 2013.