A New Data Integrity Checking Protocol with Public Verifiability in Cloud Storage

Mihir R. Gohel, Bhavesh N. Gohil

A New Data Integrity Checking Protocol with Public Verifiability in Cloud Storage

Mihir R. Gohel and Bhavesh N. Gohil

Department of Computer Engineering, NIT - Surat, Surat-395007, India
{mihirgohel, bngohil06}@gmail.com

Abstract. Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. It also eliminates their physical control of storage dependability and security, which traditionally has been expected by both enterprises and individuals. This unique paradigm brings about many new security challenges, which need to be clearly understood and resolved. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. To ensure the correctness of data, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud consumer, to verify the integrity of the data stored in the cloud. This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients.

Keywords: cloud storage, data integrity, public verifiability, Trusted Third Party Auditor (TPA).

1 Introduction

A new computing technology in today's world, called *cloud computing*, has been enabled to reality because of the rapid development of processing and storage technologies, ubiquitously available Internet, cheaper and more powerful computing resources than ever before. In this cloud computing technology, computing resources (e.g., CPU and storage) are provided as general utilities that can be leased or released by users in an on-demand fashion. In a cloud computing environment, the role of service provider is divided into two: the *infrastructure providers* who manage cloud platforms and lease resources according to a usage-based pricing model, and *service providers*, who rent resources from one or many infrastructure providers to serve the end users. The emergence of cloud computing has made a immense impact on the Information Technology (IT) industry over the past few years, where large companies such as Google, Amazon and Microsoft, IBM endeavor to provide more powerful, reliable and cost-efficient cloud platforms, and small and medium business (SMB) enterprises try to reshape their business models to gain benefits from this new paradigm.

The cloud computing offers several benefits like scalability, rapid elasticity, ubiquitous network access, rapid deployment, pay-as-you go lower cost, low cost

disaster recovery and data storage solutions. While cloud offers these advantages, it also must ensure that they get the security aspects right.

One fundamental facet of this computing model is that data is being centralized or outsourced into the cloud. From the data owners' perspective, storing data remotely in a cloud brings the new and challenging security threats to the outsourced data. Since cloud providers (CP) are separate, 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 various reasons. 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.

For benefits of their own, there are various motivations for CPs to behave unfaithfully toward Cloud Consumers regarding the status of their outsourced data. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients. Other examples include CPs, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed [1], or even hiding data loss incidents to maintain a reputation [2].

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 [1, 2]. In particular, simply downloading the data for its integrity verification is not a practical solution due to the high cost of I/O and transmission across the network. 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 expensive for data owners [1, 2]. Moreover, from the system usability point of view, data owners should be able to just use cloud storage as if it is local, without worrying about the need to verify its integrity.

In this paper, we are dealing with the problem of implementing a protocol for Public verifiable remote data integrity check, where data owners can resort to an external third party auditor (TPA) to verify the integrity of outsourced data when needed. Third party auditing provides a transparent yet cost-effective method for establishing trust between data owner and cloud server. Public verifiable remote integrity check relaxes users from the computation and online burden for periodical integrity check, especially desirable when the user is equipped with a low end computation device (e.g. smart phone, PDA) or is not always connected to the Internet.

- We propose a data integrity checking protocol for cloud storage, which can be viewed as an adaption of Sravan Kumar et al.'s protocol [3]. The proposed protocol inherits the protocol for data integrity verification, and supports public verifiability.
- The problem is further complicated by the fact that the owner of the data may be a small device, like a PDA or a mobile phone, which have limited CPU power, battery power and communication bandwidth. Hence a data integrity proof that has to be developed needs to take the above limitations into consideration. The scheme should be able to produce a proof without the need for the server to access the entire file or the client retrieving the entire file from the server. Also the scheme should minimize the local computation at the client as well as the bandwidth consumed at the client.

2 Related Work

The simplest Proof of data integrity can be made using a keyed hash function h_k(F). In this scheme the Cloud Consumer (CC), before archiving the data file F in the cloud storage server (CSS), pre-computes the cryptographic hash of F using h_k(F) and stores this hash as well as the secret key K. CC transfers this key and pre-computed hash value to Trusted Third Party Auditor (TPA), to verify the integrity of the file F at regular interval. To check if the integrity of the file F is lost the TPA releases the secret key K to the cloud archive and asks it to compute and return the value of h_k(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). At 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 [3].

Recently, much of growing interest has been pursued in the context of remotely stored data verification [1–5]. Zhang and Chen have proposed an Integrity check scheme based on well-known RSA Security assumption called A RSA-based Assumption Data Integrity Check without Original Data [4]. In which they uses the concept of Random Oracle Model and RSA to verify the intactness of data. The proposed scheme is proven to be secure in Random oracle model.

Surya et al. have also proposed a protocol for the same called Data Integrity as a Service (DIaaS) [5]. The proposed protocol needs to have complex infrastructure to be implemented. i.e., Trust Management Service (TMS), Cloud Storage Service (CSS), Key Management Service (KMS) and Integrity Management Service (IMS). The proposed protocol also performs more no. of encryptions and hashing to verify the integrity.

Ari Juels and Burton S. Kaliski Jr. proposed a scheme called Proof of retrievability for large files using ”sentinels” [1]. 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 retrievability it wants to verify. In this scheme special blocks (called sentinels) are hidden among other blocks in the data file F. To make the sentinels indistinguishable from the data blocks, the whole modified file is encrypted and stored at the archive. As this scheme involves the encryption of the file F using a secret key it becomes 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.) [3]. The schematic view of this approach is shown in Figure 1.

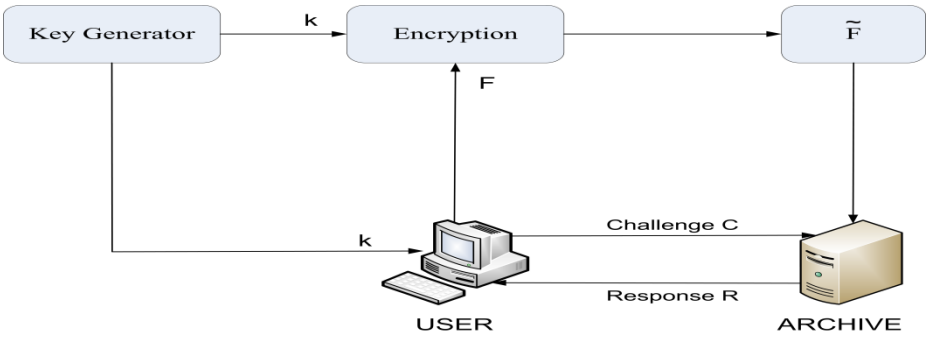

Fig. 1. Schematic views of a proof of retrievability based on inserting random sentinels in the file F [1]

Sravan Kumar R and Ashutosh Saxena have proposed scheme for data integrity proof [3], which does not involve the encryption of the whole data but encrypts only few bits of data per data Block, thus reducing the computational overhead on the clients. The client storage overhead is also minimized as it does not store any data with it. So the scheme suits well for thin clients. But the proposed scheme restricts the remote data verifiability to private only. It doesn't allow any third party auditor to verify the integrity of data, on behalf of client. The clients themselves need to devote their computation resources to perform frequent integrity checks. Hence, this scheme burdensome the client.

3 The Proposed Data Integrity Checking Protocol in Cloud Storage with Public Verifiability

In our proposed data integrity protocol, we inherit the support of data integrity from [3], and support of TPA from Qian Wang et al. [6]. In our proposed data integrity protocol, the client doesn't need to store any data with it. Verifier needs only a single cryptographic key and two functions which generate a random sequence. The client before storing the file at the archive, preprocesses the file and appends some meta data to the file and stores at the archive. The client then transfers the key and functions to the TPA to audit the file frequently. At the time of verification the TPA uses this meta data to verify the correctness of data. Our proposed scheme neither prevents the archive from modifying or deletions data nor preserves data privacy against TPA. Representative network architecture for cloud data storage with TPA is illustrated in Fig. 2.

3.1 Setup phase

Let the client C wishes to the store the file F with the archive. Let this file F consist of n file blocks. We initially preprocess the file and create metadata to be appended to the file. Let each of the n data blocks have m bits in them. The initial setup phase is represented in Fig 3 and can be described in the following steps:

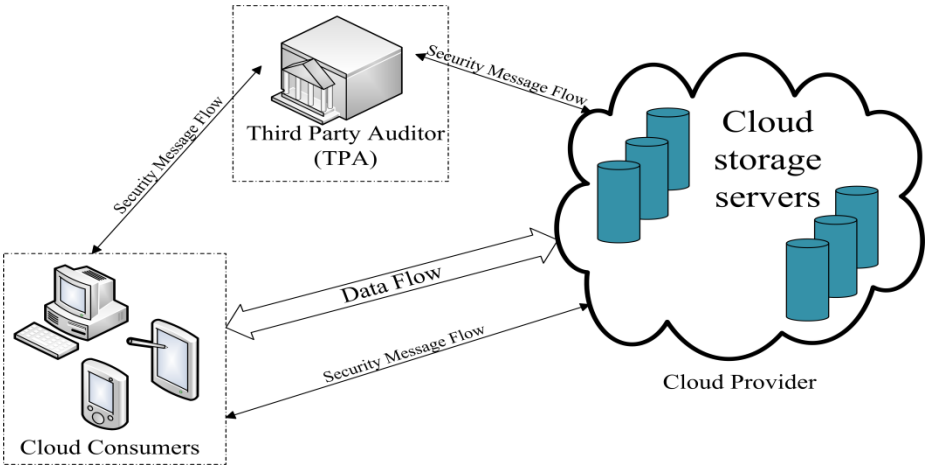

Fig. 2. Cloud Data Storage Architecture with TPA [6].

1) Generation of metadata: Let g be the function defined as

$$g(i,j) \rightarrow \{1..m\}, i \epsilon \{1..n\}, j \epsilon \{1..k\} \quad (1)$$

Where k is the number of bits per data block which we wish to read as meta data. The function g generates for each data block a set of k bit positions within the m bits that are in the data block. Hence $g(i,\ j)$ gives the j^{th} bit in the i^{th} data block. The value of k is in the choice of the client and is a secret known only to him. Therefore for each data block we get a set of k bits and in total for all the n blocks we get $n*k$ bits. Let mi represent the k bits of meta data for the i^{th} block.

2) Encrypting the meta data: Each of the meta data from the data blocks m_i is encrypted by using a suitable algorithm to give a new modified meta data M_i. Without loss of generality we show this process by using a simple XOR operation. Let h be a function which generates a k bit integer α_i for each i.

$$h : i \longrightarrow \alpha_i, \alpha_i \epsilon \{0 .. 2^k\} \quad (2)$$

For the meta data (m_i) of each data block the number α_i is added to get a new k bit number M_i.

$$M_i = m_i + \alpha_i \quad (3)$$

In this way we get a set of n new meta data bit blocks.

3) Appending of meta data: All the meta data bit blocks that are generated using the above procedure are to be concatenated together. This concatenated meta data should be appended to the file F before storing it at the cloud server. The file F along with the appended meta data $\check{F}$ is archived with the cloud.

4) Transferring the meta data attributes to TPA: Client now transfers the two functions g and h to trusted TPA. TPA audits the file F archived with cloud by using these two functions g and h at the time of data integrity verification.

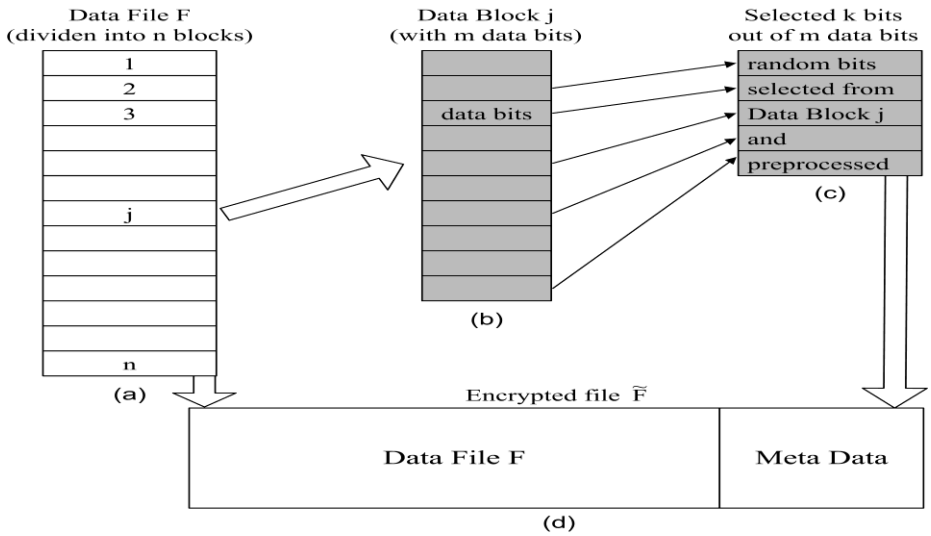

Fig. 3. (a) A data file F with n data blocks (b) a data block j, having m bits, is selected to preprocess (c) randomly selected k bits outcome of function g (d) preprocessed k bits are appended to data file F, the encrypted file $\tilde{F}$ will be stored at cloud.

3.2 Verification phase

Let the TPA want to verify the integrity of the file F. It throws a challenge to the archive and asks it to respond. The challenge and the response are compared and the TPA accepts or rejects the integrity proof. Suppose the TPA wishes to check the integrity of n^{th} block. The TPA challenges the cloud storage server by specifying the block number i and a bit number j generated by using the function g. The TPA also specifies the position at which the meta data corresponding the block i is appended. This meta data will be a k bit number. Hence the cloud storage server is required to send $k + 1$ bits for verification by the client. The meta data sent by the cloud is decrypted by using the Number $\propto_i$ and the corresponding bit in this decrypted meta data is compared with the bit that is sent by the cloud. Any mismatch between the two would mean a loss of the integrity of the clients' data at the cloud storage.

4 Conclusions and Future Works

To ensure cloud data storage security, it is critical to enable a third party auditor (TPA) to evaluate the service quality from an objective and independent perspective. Public verifiability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications.

In this paper we have worked to facilitate the client in getting a proof of integrity of the data. Our proposed scheme is developed to reduce the computational and storage overhead of the client with public verifiability as well as to minimize the compu-

tational 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 [1-2, 4-5] that were developed. Hence this scheme proves advantageous to thin clients like PDAs and mobile phones.

Our scheme applies only to static storage of data. If archived file modifies dynamically, then client has to preprocess the file each time he modifies the file. Also scheme doesn't preserve privacy of data against TPA. Hence developing on this will be a future challenge.

References

1. A. Juels, J. Burton, and S. Kaliski, "PORs: Proofs of Retrievability for Large Files," in *Proc. ACM CCS '07*, pp. 584–97, Oct. 2007.
2. G.Ateniese et al., "Provable Data Possession at Untrusted Stores," in *Proc. ACM CCS '07*, pp.598–609, Oct. 2007.
3. Sravan Kumar, Saxena A, "Data integrity proofs in cloud storage," in *proc. COMSNETS'11*, pp.1-4, Jan.2011.
4. Zhang Jianhong, Chen Hua , "Secuirty storage in the Cloud Computing: A RSA-based assumption data integrity check without original data," in *Proc. ICEIT'10*, pp. 17-19, Sept. 2010.
5. Nepal S., Shiping Chen, Jinhui Yao, Thilakanathan D., "DIaaS: Data Integrity as a Service in the Cloud," in *Proc. IEEE Cloud Computing (CLOUD)'11*, pp.308-315, July.2011.
6. Qian W, Cong W, Jin L, *et al.,* "Enabling public verifiability and data dynamics for storage security in cloud computing". *Lecture Notes in Computer Science*, 2009, 5789: 355–370.