Efficient Audit Service Outsourcing for Data Integrity in Clouds

P. RAMACHANDRA RAO
M.Tech, Progressive Engineering College, Hyderabad, AP-INDIA. E-mail: ursramachandra@gmail.com.

S. CHAKRADHAR GOUD
Assoc Prof, Progressive Engineering College, Hyderabad, AP-INDIA. E-mail: cgsakki@yahoo.com.

Abstract: Cloud-based outsourced storage relieves the client’s burden for storage management and maintenance by providing a comparably low-cost, scalable, location-independent platform. However, the fact that clients no longer have physical possession of data indicates that they are facing a potentially formidable risk for missing or corrupted data. To avoid the security risks, audit services are critical to ensure the integrity and availability of outsourced data and to achieve digital forensics and credibility on cloud computing. Provable data possession (PDP), which is a cryptographic technique for verifying the integrity of data without retrieving it at an untrusted server, can be used to realize audit services. In this paper, profiting from the interactive zero-knowledge proof system, we address the construction of an interactive PDP protocol to prevent the fraudulence of prover (soundness property) and the leakage of verified data (zero-knowledge property). We prove that our construction holds these properties based on the computation Diffie–Hellman assumption and the rewindable black-box knowledge extractor. We also propose an efficient mechanism with respect to probabilistic queries and periodic verification to reduce the audit costs per verification and implement abnormal detection timely. In addition, we present an efficient method for selecting an optimal parameter value to minimize computational overheads of cloud audit services. Our experimental results demonstrate the effectiveness of our approach.

Keywords: Security Cloud storage, Interactive proof system, Provable data possession, Audit service.

I. INTRODUCTION

In recent years, the emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional information technology. Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. One fundamental aspect of this paradigm shifting is that data are being centralized and outsourced into clouds. This kind of outsourced storage services in clouds have become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients’ data. The cloud storage service (CSS) relieves the burden of storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to users since their data or archives are stored into an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices.

However, they are still susceptible to security threats both from outside and inside the cloud (Armbrust et al., 2010); for the benefits of their possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users (Tchilifionova, 2011); furthermore, the dispute occasionally suffers from the lack of trust on CSP. Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users’ own improper operations (Ko et al., 2011). Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of the stored data (Yavuz and Ning, 2009).

Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature schemes (Hsiao et al., 2009; Yumerefindi and Chase, 2007), cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive transaction, especially for large-size files. Moreover, the solutions to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users (Armbrust et al., 2010). Therefore, it is crucial to realize public auditability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and data assurance in clouds.

Copyright © 2013 SEMAR GROUPS TECHNICAL SOCIETY. All rights reserved.
To implement public auditability, the notions of proof of retrievability (POR) (Juels, 2007) and provable data possession (PDP) (Ateniese et al., 2007) have been proposed by some researchers. Their approach was based on a probabilistic proof technique for a storage provider to prove that clients’ data remain intact without downloading the stored data, which is called “verification without downloading”. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, this provides us an effective approach to accommodate the requirements from public auditability. POR/PDP schemes evolved around an untrusted storage offer a publicly accessible remote interface to check the tremendous amount of data.

Although PDP/POR schemes evolved around untrusted storage offer a publicly accessible remote interface to check and manage tremendous amount of data, most of existing schemes cannot give a strict security proof against the untrusted CSP’s deception and forgery, as well as information leakage of verified data in verification process. These drawbacks greatly affect the impact of cloud audit services. Thus, new frameworks or models are desirable to enable the security of public verification protocol in cloud audit services.

Another major concern addressed by this paper is how to improve the performance of audit services. The audit performance concerns not only the costs of computation, communication, storage for audit activities but also the scheduling of audit activities. No doubt improper scheduling, more or less frequent, causes poor audit performance, but an efficient scheduling can help provide a better quality of and a more cost-effective service. Hence, it is critical to investigate an efficient schedule for cloud audit services.

In response to practical requirements for outsourced storages, our concerns to improve the performance of audit services are mainly from three aspects:

- How to design an efficient architecture of audit system to reduce the storage and network overheads and enhance the security of audit activities.
- How to provide an efficient audit scheduling to help provide a more cost-effective audit service.
- How to optimize parameters of audit systems to minimize the computation overheads of audit services.

Solving these problems will help to improve the quality of audit services, which can not only timely detect abnormality, but also take up fewer resources, or rationally allocate resources.

A. Contributions

In this paper, we focus on efficient audit services for outsourced data in clouds, as well as the optimization for high-performance audit schedule. First of all, we propose architecture of audit service outsourcing for verifying the integrity of outsourced storage in clouds. This architecture based on cryptographic verification protocol does not need to trust in storage server providers. Based on this architecture, we have made several contributions to cloud audit services as follows:

- We provide an efficient and secure cryptographic interactive audit scheme for public auditability. We prove that this scheme retains the soundness property and zero-knowledge property of proof systems. These two properties ensure that our scheme can not only prevent the deception and forgery of cloud storage providers, but also prevent the leakage of outsourced data in the process of verification.
- We propose an efficient approach based on probabilistic queries and periodic verification for improving performance of audit services. To detect abnormal situations timely, we adopt a way of sampling verification at appropriate planned intervals.
- We presented an optimization algorithm for selecting the kernel parameters by minimizing computation overheads of audit services. Given the detection probability and the probability of sector corruption, the number of sectors has an optimal value to reduce the extra storage for verification tags, and to minimize the computation costs of CSPs and clients’ operations.

In practical applications, above conclusions will play a key role in obtaining a more efficient audit schedule. Further, our optimization algorithm also supports an adaptive parameter selection for different sizes of files (or clusters), which could ensure that the extra storage is optimal for the verification process.

Finally, we implement a prototype of an audit system to evaluate our proposed approach. Our experimental results not only validate the effectiveness of above-mentioned approaches and algorithms, but also show our system has a lower computation cost, as well as a shorter extra storage for verification. We list the features of our PDP scheme in Table 1. We also include a comparison of related techniques, such as, PDP (Ateniese et al., 2007),
DPDP (Erway et al., 2009), and CPOR (Shacham and Waters, 2008). Although the computation and communication overheads of O(t) and O(1) in PDP/SPDP schemes are lower than those of O(t + s) and O(s) in our scheme, our scheme has less complexity due to the introduction of a fragment structure, in which an outsourced file is split into n blocks and each block is also split into s sectors. This means that the number of blocks in PDP/SPDP schemes is s times more than that in our scheme and the number of sampling blocks t in our scheme is merely 1/s times more than that in PDP/SPDP schemes. Moreover, the probability of detection in our scheme is much greater than that in PDP/SPDP schemes because of \[1 - (1 - \rho_s)^s \geq 1 - (1 - \rho_p)^t\]. In addition, our scheme, similar to PDP and CPOR schemes, provides the ownership proof of outsourced data as a result that it is constructed on the public-key authentication technology, but SPDP and DPDP schemes cannot provide such a feature because they are only based on the Hash function.

**B. Organization**

This paper is organized as follows: in Section 1, we describe an audit architecture and security requirements of cloud audit systems. Section 2 introduces our audit scheme and analyzes the security of our scheme. In Section 3, we analyze the audit performance based on probabilistic queries and periodic verification. Section 4 gives an optimization algorithm of tag storage and verification protocol. Our implementation and experimental results are described in Section 5. Section 6 overviews the related work and we conclude this paper in Section 7.

### II. AUDIT SYSTEM ARCHITECTURE

In this section, we first introduce an audit system architecture for outsourced data in clouds in Fig. 1, which can work in an audit service outsourcing mode. In this architecture, we consider a data storage service containing four entities:

- **Data owner (DO):** who has a large amount of data to be stored in the cloud?
- **Cloud service provider (CSP):** who provides data storage service and has enough storage spaces and computation resources;
- **Third party auditor (TPA):** who has capabilities to manage or monitor outsourced data under the delegation of data owner; and
- **Granted applications (GA):** who have the right to access and manipulate stored data. These applications can be either inside clouds or outside clouds according to the specific requirements.

Next, we describe a flowchart for audit service based on TPA. This also provides a background for the description of our audit service outsourcing as follows:

- First, the client (data owner) uses the secret key sk to preprocesses the file, which consists of a collection of n blocks, generates a set of public verification information that is stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>CSP computation</th>
<th>Client computation</th>
<th>Communication</th>
<th>Fragment structure</th>
<th>Privacy</th>
<th>Ownership proof</th>
<th>Prob. of detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDP (Ateniese et al., 2007)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>SPDP (Ateniese et al., 2008)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>DPDP-I (Erway et al., 2009)</td>
<td>O(1 + s)</td>
<td>O(1 + s)</td>
<td>O(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>DPDP-II (Erway et al., 2009)</td>
<td>O(1 + s)</td>
<td>O(1 + s)</td>
<td>O(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>CPOR-I (Shacham and Waters, 2008)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>CPOR-II (Shacham and Waters, 2008)</td>
<td>O(1 + s)</td>
<td>O(1 + s)</td>
<td>O(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
<tr>
<td>Our scheme</td>
<td>O(1 + s)</td>
<td>O(1 + s)</td>
<td>O(1 + s)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 - (1 - \rho_p)^t</td>
</tr>
</tbody>
</table>

**Table 1: Comparison of POR/PDP schemes for a file consisting of n blocks**

- Note that s is the number of sectors in each block, t is the number of sampling blocks, and \(\rho_p, \rho_s\) are the probability of block and sector corruption in a cloud server, respectively.

At a later time, using a protocol of proof of retrievability, TPA (as an audit agent of clients) issues a challenge to audit (or check) the integrity and availability of the outsourced data in terms of the public verification information. It is necessary to give an alarm for abnormal events.

This architecture is known as the audit service outsourcing due to data integrity verification can be implemented by TPA without help of data owner. In this architecture, the data owner and granted clients need to dynamically interact with CSP to access or update their data for various application purposes. However, we...
neither assume that CSP is trust to guarantee the security of stored data, nor assume that the data owner has the ability to collect the evidences of CSP’s fault after errors occur. Hence, TPA, as a trust third party (TTP), is used to ensure the storage security of their outsourced data. We assume the TPA is reliable and independent, and thus has no incentive to collude with either the CSP or the clients during the auditing process:

- TPA should be able to make regular checks on the integrity and availability of these delegated data at appropriate intervals;
- TPA should be able to take the evidences for the disputes about the inconsistency of data in terms of authentic records for all data operations.

In this audit architecture, our core idea is to maintain the security of TPA to guarantee the credibility of cloud storages. This is because it is more easy and feasible to ensure the security of one TTP than to maintain the credibility of the whole cloud. Hence, the TPA could be considered as the root of trust in clouds. To enable privacy-preserving auditing for cloud data storage under this architecture, our protocol design should achieve following security and performance guarantees:

**Audit-without-downloading:** To allow TPA (or other clients with the help of TPA) to verify the correctness of cloud data on demand without retrieving a copy of whole data or introducing additional on-line burden to the cloud users.

**Verification-correctness:** To ensure there exists no cheating CSP that can pass the audit from TPA without indeed storing users’ data intact.

**Privacy-preserving:** to ensure that there exists no way for TPA to derive users’ data from the information collected during the auditing process. And

**High-performance:** to allow TPA to perform auditing with minimum overheads in storage, communication and computation, and to support statistical audit sampling and optimized audit schedule with a long enough period of time.

To support this architecture, a cloud storage provider only needs to add a corresponding algorithm module to implement this audit service. Since the audit process could be considered as an interactive protocol implementation between TPA and this module, such a module is usually designed as a server daemon to respond audit requests of TPA through cloud interfaces. This daemon is just a simple lightweight service due to the reason that it does not need to transfer the verified data to the TPA (audit-without-downloading property). Hence, this daemon can be easily appended into various cloud computing environments.

### III. ACTIVITIES OF THE CLOUD

**A. Devoted Property Assertion**

All resources have high-quality assurance in its multi cloud servers with the help of strong platform. The mentioned possessions of RAM, CPU, and Bandwidth capacity [4][5] in network produce sufficient atmospheres for its concerned usage. Thus the activity of could server can’t ever opposite to the current world while considering its level of integrity and security even though there is
some little bit of conflict that we want yet to recover from its already existing level.

**B. Excess Data Storage**

All cloud server storage resources are managed by high-performance and high-availability storage area network. Many cloud solutions operate on local disks from the host system, which means any computing or storage failure can result in down time and potential data loss. As cloud servers are autonomous, if there happens any server crack in stored data, these can be protected against internal and external attacks.

**C. Client Support and Stability**

It clearly tells us that user can do the increase and decrease of the data capacity in the cloud server with the help of CSP (cloud service provider) in his request. This storage level must be with flexible and durability condition as far as its entire design or structure is concerned. Thus it should be claimed extra storage space concerning future process in data exchange.

**D. Pay Costs for Access**

Customers have to be charged according to their usage and their data storage in cloud storage apart from allocated space. Customers can charge depending on the usage of network access for data exchange with good bandwidth [4] [5] as well as data storage. For example, it is just like when utility company sells power to consumers and telephonic company offering utilities for their services.

**E. Efficient Dynamic Computing Environment**

Data should be in a self-motivated computing infrastructure. The main concept in this dynamic environment is that all standardized and scalable infrastructure should have dynamic operation such as modification, append, and delete. The cloud platform which has virtualized conditions also should have some specific independent environment.

**F. Accurate Choice of CSP**

To get excellent service from multiple servers, good service providers are important to be considered and selected. So much care must be taken in this respect so that the CSP itself can be elastic with clients in order to get accessed with all places (anywhere and anytime). It has the following benefits such as, 1) Cost savings to save expenditure among IT capabilities. 2) Trustworthiness- Data back up by CSP in cloud servers if system is stolen and loses the data itself. 3) Scalability on requirement-

Whenever there is a need for data accessing, user can be easily accessible to that anytime and anywhere. 4) Protection expertise- Cloud service providers in general have more skill securing networks than individuals and IT personnel whose networks are usually associated to the public Internet anyway. 5) All over Access- It allows users ubiquitous access to computing power, storage, and applications.

**IV. KEY TECHNOLOGIES**

**A. Essential elements of Cloud Computing**

The essential elements of cloud computing are as follows

**On-demand self-service:** A consumer with an instantaneous need at a particular timeslot can avail computing resources in an automatic rashion without resorting to human interactions with providers of these resources.

**Broad network access:** These computing resources are delivered over the network and used by various client applications with heterogeneous platforms situated ata a consumer’s site.

**Resource pooling:** A cloud service provider’s computing resources are pooled together in an effort to serve multiple consumers using either the multi-tenancy or the virtualization model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. For example, consumers are not able to tell where their data is going to be stored in the cloud.

**Rapid elasticity:** For consumers, computing resources become immediate rather than persistent, there are no upfront commitment and contract as they can use them to scale up whenever they want, and release them once they finish scaling down. Moreover, resources provisioning appears to be infinite to them, the consumption can rapidly rise in order to meet peak requirement at any time.

**B. Deployment models**

There are three deployment models for cloud computing: public, private and hybrid cloud.

**Public cloud:** The public cloud is used by the general public cloud consumers and the cloud service providers have the full ownership of the public cloud with is own policy, value, and profit, costing and charging model. The services are made available to the users through the Internet. Eg: Amazon EC2, S3, Google AppEngine.
Private cloud: The cloud infrastructure is operated solely within a single organization and managed by the organization or a third party regardless whether it is located premise or off premise. The motivation to setup a private cloud within organization has several aspects. First, to maximize and optimize the utilization of existing resources. Second, security concerns including data privacy and trust also make private cloud an option for many firms. Third, data transfer cost from local IT infrastructure to a Public cloud still rather considerable. Fourth, organizations always require full control over mission-critical activities that reside behind their firewalls.

Community cloud: Several organizations jointly constructs and share the same cloud infrastructure as well as policies, requirements, values and concerns. The cloud community forms into a degree of economic scalability and democratic equilibrium. The cloud infrastructure could be hosted by a third party vendor or within one of the organizations in the community.

Hybrid cloud: It is a combination of both private and public cloud which makes data and application portability.

C. Service models

There are various kinds of service models available in cloud namely

1. **Software as a Service (SaaS):** SaaS provides various applications as service that can be accessed through the network by users and deployed by people those who are using cloud.

   Eg: Google mail, Google docs, SalesForce.com.

2. **Platform as a Service(PaaS):** PaaS provides the platform to develop the application. The major difference between the Paas and SaaS is that SaaS only host the completed application but PaaS offers a development platform.

   Eg: Google AppEngine.

3. **Infrastructure as a Service (IaaS):** In IaaS, the cloud consumers can directly use IT infrastructures such as processing, storage, network and other fundamental computing resources [7].

   Eg: Amazon EC2.

D. Cloud DBMS Features

**Efficiency:** Given that the cloud computing pricing is structured in a way so that you pay for only what you use, the price increases linearly with the requisite storage, network bandwidth and compute power. Hence, if data analysis software product A requires an order of magnitude more compute units than data analysis software product B to perform the same task, then product A will cost an order of magnitude more than B. Efficient software has a direct effect if the bottom line.

**Fault Tolerance:** Fault tolerance in the context of analytical data workloads is measured differently than fault tolerance in the context of transactional workloads. For transactional workloads, a fault tolerant DBMS can recover from a failure without losing any data or updates from recently committed transactions and in the context of distributed databases, can successfully commit transactions and make progress on a workload even in the face of worker node failure. For read only queries in analytical workloads, there are no right transactions to commit, nor updates to lose upon node failure. Hence, a fault tolerant analytical DBMS is simple one that does not have to restart a query if one of the nodes involved in query processing.

**Ability to run in heterogeneous environment:** The performance of cloud compute nodes is often not consistent, with some nodes attaining orders of magnitude worse performance than other nodes. There are a variety of reasons why this could occur, ranging from hardware failure causing degraded performance on a node, to an instance being unable to access the second core on a dual core machine, to contention for non-virtualized resources. If the amount of work needed to execute a query is equally divided amongst the cloud compute nodes, then there is danger that the time to complete the query will be approximately equal to time for the slowest compute node to complete its assigned task. A node observing degraded performance would thus have a disproportionate affect of total query latency. A system designed to run in a heterogeneous environment would take appropriate measures to prevent this from occurring. A system designed to run in a heterogeneous environment would take appropriate measures to prevent this from occurring.

**Ability to operate on encrypted data:** Sensitive data may be encrypted before being uploaded to the cloud. In order to prevent unauthorized access to the sensitive data, any application running in the cloud should not have the ability to directly decrypt the data before accessing it. However, whipping entire tables or columns out of the cloud for decryption is bandwidth intensive. Hence, the ability of the data analysis system to operate directly on encrypted data so that a smaller amount of data needs to ultimately be shipped elsewhere to be decrypted could significantly improve performance.
E. Provable Data Possession (PDP)

We describe a framework for provable data possession. This provides background for related work and for the specific description of our schemes. A PDP protocol checks that an outsourced storage site retains a file, which consists of a collection of n blocks. The client (data owner) pre-processes the file, generating a piece of metadata that is stored locally, transmits the file to the server, and may delete its local copy. The server stores the file and responds to challenges issued by the client. Storage at the server is in $\Omega(n)$ and storage at the client is in $O(1)$, conforming to our notion of an outsourced storage relationship. As part of pre-processing, the client may alter the file to be stored at the server. The client may expand the file or include additional metadata to be stored at the server. Before deleting its local copy of the file, the client may execute a data possession challenge to make sure the server has successfully stored the file. The client may encrypt a file prior to outsourcing the storage. For our purposes, encryption is an orthogonal issue; the “file” may consist of encrypted data and our metadata does not include encryption keys. The client requests that the server compute a function of the stored file, which it sends back to the client. Using its local metadata, the client verifies the response [6].

Threat model: The server S must answer challenges from the client C; failure to do so represents a data loss. However, the server is not trusted: Even though the file is totally or partially missing, the server may try to convince the client that it possesses the file. The server’s motivation for misbehaviour can be diverse and includes reclaiming storage by discarding data that has not been or is rarely accessed (for monetary reasons), or hiding a data loss incident (due to management errors, hardware failure, compromise by outside or inside attacks etc). The goal of a PDP scheme that achieves probabilistic proof of data possession is to detect server misbehaviour when the server has deleted a fraction of the file. Zero knowledge property prevents the leakage of verified data and the soundness property prevents the fraudulence of the prover.

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

To validate the efficiency of our approach, we have implemented a prototype of an audit system based on our proposed solution. This system has been developed in an experimental cloud computing system environment (called M-Cloud) of Peking University, which is constructed within the framework of the IaaS to provide powerful virtualization, distributed storage, and automated management. To verify the performance of our solution, we have simulated our audit service and storage service using two local IBM servers with two Intel Core 2 processors at 2.16 GHz and 500M RAM running Windows Server 2003 and 64-bit Redhat Enterprise Linux Server 5.3, respectively. These two servers were connected into the Mccloud via 250 MB/s of network bandwidth. The storage server was responsible for managing a 16TB storage array based on Hadoop distributed file system (HDFS) 0.20 clusters with 8 worker nodes located in our laboratory.

To develop the TPA’s schedule algorithm and CSP’s verification daemon, we have used the GMP and PBC libraries to implement a cryptographic library. This C library contains approximately 5200 lines of codes and has been tested on Windows and Linux platforms. The elliptic curve utilized in the experiment is a MNT curve, with base field size of 160 bits and the embedding degree 6. The security level is chosen to be 80 bits, which means $|p| = 160$.

Fig.2. Experiment results under different file size, sampling ratio, and sector number.
Firstly, we quantify the performance of our audit scheme under different parameters, such as file size $sz$, sampling ratio $w$, sector number per block $s$, and so on. Our previous analysis shows that the value of $s$ should grow with the increase of $sz$ in order to reduce computation and communication costs. Thus, our experiments were carried out as follows: the stored files were chosen from 10 KB to 10 MB, the sector numbers were changed from 20 to 250 in terms of the file sizes, and the sampling ratios were also changed from 10% to 50%. The experimental results were showed in the left side of Fig. 2. These results dictate that the computation and communication costs (including I/O costs) grow with increase of file size and sampling ratio.

Next, we compare the performance for each activity in our verification protocol. We have described the theoretical results: the overheads of “commitment” and “challenge” resemble one another, and the overheads of “response” and “verification” also resemble one another. To validate the theoretical results, we changed the sampling ratio $w$ from 10% to 50% for a 10 MB file and 250 sectors per block. In the right side of Fig. 2, we show the experiment results, in which the computation and communication costs of “commitment” and “challenge” are slightly changed for sampling ratio, but those for “response” and “verification” grow with the increase of sampling ratio.

Finally, we evaluate the performance of our audit scheme in terms of computational overhead; due to these two schemes have constant-size communication overhead. For sake of comparison, our experiments used the same scenario as previous analysis, where a fix-size file is used to generate the tags and prove possession under different $s$. For a 150 kB file, the computational overheads of verification protocol are showed in Fig. 3(a) when the value of $s$ ranges from 1 to 50 and the size of sector is 20-Bytes. It is obvious that the experiment result is consistent with the analysis. The computational overheads of the tag generation are also showed in Fig.3(b) in the same case. The results indicate that the overhead is reduced when the value of $s$ is increased.

However there is a considerable amount of work done on untrusted outsourced storage. The most direct way to enforce the integrity control is to employ cryptographic hash function. Yumerefendi and Chase proposed a solution for authenticated network storage (Yumerefendi and Chase, 2007; Hsiao et al., 2009), using a hash tree (called as Merkle tree) as the underlying data structure.

VI. RELATED WORKS

There has been a considerable amount of work done on untrusted outsourced storage. The most direct way to enforce the integrity control is to employ cryptographic hash function. Yumerefendi and Chase proposed a solution for authenticated network storage (Yumerefendi and Chase, 2007; Hsiao et al., 2009), using a hash tree (called as Merkle tree) as the underlying data structure.

However their processing of updates is computationally expensive. Fu et al. (2002) described and implemented a method for efficiently and securely accessing a read-only file system that has been distributed to many providers. This architecture is a solution for efficiently authenticating operations on an outsourced file system.

Some recent work (Li et al., 2006; Ma et al., 2005; Xie et al., 2007; Yavuz and Ning, 2009) studied the problem of auditing the integrity for outsourced data or database. By explicitly assuming an order of the records in database, Pang et al. (Ma et al., 2005) used an aggregated signature to sign each record with the information from two neighboring records in the ordered sequence, which ensures the result of a simple selection query is continuous by checking the aggregated signature. Other work (Li et al., 2006; Xie et al., 2007) used a Merkle tree to audit the completeness of query results, but in some extreme cases, the overhead could be as high as processing these queries locally, which can significantly undermine the benefits of database outsourcing. Moreover, to ensure freshness, an extra system is needed to deliver the up-to-date root signature to all clients in a reliable and timely manner.
To check the integrity of stored data without download, some researchers have proposed two basic approaches called provable data possession (PDP) (Ateniese et al., 2007) and proofs of retrievability (POR) (Juels, 2007). Ateniese et al. (2007) first proposed the PDP model for ensuring possession of files on untrusted storages and provided an RSA-based scheme for the static case that achieves O(1) communication costs. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the servers for data possession. This property greatly extend application areas of PDP protocol due to the separation of data owners and the authorized users.

In order to support dynamic data operations, Ateniese et al. have developed a dynamic PDP solution called scalable PDP (Ateniese et al., 2008). They proposed a lightweight PDP scheme based on cryptographic Hash function and symmetric key encryption, but the servers can deceive the owners using previous metadata or responses due to lack of the randomness in the challenge. The number of updates and challenges is limited and fixed a priori. Also, one cannot perform block insertions anywhere. Based on this work, Erway et al. (2009) introduced two dynamic PDP schemes with a Hash function tree to realize O(log n) communication and computational costs for a file consisting of n blocks. The basic scheme, called DPDP-I, remains the drawback of SPDP, and in the ‘blockless’ scheme, called DPDP-II, the data blocks can be leaked by the response of challenge.

Juels (2007) presented a POR scheme which relies largely on preprocessing steps the client conducts before sending a file to CSP. Unfortunately, these operations prevent any efficient extension to update data. Shacham and Waters (2008) proposed an improved version of this protocol called Compact POR, which uses homomorphic property to aggregate a proof into O(1) authenticator value and O(t) computation costs for t challenge blocks, but their solution is also static and there exist leakages of data blocks in the verification process. Wang et al. (2009) presented a dynamic scheme with O(log n) costs by integrating above CPOR scheme and Merkle Hash Tree (MHT) in DPDP. Furthermore, several POR schemes and models have been recently proposed including (Bowers et al., 2009; Dodis et al., 2009). Since the responses of challenges have homomorphic property, above schemes (especially CPOR schemes) can leverage the PDP construction in hybrid clouds.

Based on above works, Wang et al. (2010) introduced PDP/POR schemes into audit systems. This work is motivated by the public audit systems of data storages and provided a privacy-preserving auditing protocol. Moreover, this scheme achieves batch auditing to support efficient handling of multiple auditing tasks. Although their solution is not suitable for practical applications because of lack of support for dynamic operations and rigorous performance analysis, it points out a promising research direction for checking the integrity of outsourced data in untrusted storage.

VII. CONCLUSIONS

In this paper, we addressed the construction of an efficient audit service for data integrity in clouds. Profiting from the standard interactive proof system, we proposed an interactive audit protocol to implement the audit service based on a third party auditor. In this audit service, the third party auditor, known as an agent of data owners, can issue a periodic verification to monitor the change of outsourced data by providing an optimized schedule. To realize the audit model, we only need to maintain the security of the third party auditor and deploy a lightweight daemon to execute the verification protocol. Hence, our technology can be easily adopted in a cloud computing environment to replace the traditional Hash-based solution.

More importantly, we proposed and quantified a new audit approach based on probabilistic queries and periodic verification, as well as an optimization method of parameters of cloud audit services. This approach greatly reduces the workload on the storage servers, while still achieves the detection of servers’ misbehavior with a high probability. Our experiments clearly showed that our approach could minimize computation and communication overheads.

VIII. REFERENCES


