IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 5, Oct-Nov, 2013
ISSN: 2320 - 8791
www.ijreat.org

IMPROVING DATA INTEGRITY USING BILINEAR PAIRING

T.Sivasankari[1] and V.Vaishnavee[2]

[1]**Department of Computer Science and Engineering, SNS College of Engineering, Coimbatore.**

[2]**Department of Computer Science and Engineering, SNS College of Engineering, Coimbatore.**

Abstract

Provable data possession (PDP) is a technique for ensuring the integrity of data in storage outsourcing. In this paper, we address the construction of an efficient PDP scheme for distributed cloud storage to support the scalability of service and data migration, in which we consider the existence of multiple cloud service providers to cooperatively store and maintain the clients' data.We present a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy. We prove the security of our scheme based on multiprover zero-knowledge proof system, which can satisfy completeness, knowledge soundness, and zero-knowledge properties. In addition, we articulate performance optimization mechanisms for our scheme, and in particular present an efficient method for selecting optimal parameter values to minimize the computation costs of clients and storage service providers. Our experiments show that our solution introduces lower computation and Communication overheads in comparison with non cooperative approaches.

Keywords: Storage security, provable data possession, interactive protocol, zero-knowledge, multiple cloud, cooperative.

1. Introduction

IN recent years, cloud storage service has become a fasterprofit growth point by providing a comparably low cost, scalable, position-independent platform for clients' data.Since cloud computing environment is constructed based on open architectures and interfaces; it has the capability to incorporate multiple internal and/or external cloud services together to provide high interoperability. We call such a distributed cloud environment as a multi Cloud (or hybrid cloud). Often, by using virtual infrastructure management (VIM) [1], a multicloud allows clients to easily access his/her resources remotely through interfaces such as web services provided by Amazon EC2.There exist various tools and technologies for multicloud,such as Platform VM Orchestrator, VMware vSphere, and Ovirt. These tools help cloud providers construct a distributed cloud storage platform (DCSP) for managing clients' data. However, if such an important platform is vulnerable to security attacks, it would bring irretrievable losses to the clients. For example, the confidential data

Scheme	Type	CSP Comp.	Client Comp.	Comm.	Frag.	Privacy	Multiple Clouds	Prob. of Detection
PDP[2]	HomT	$O(t)$	$O(t)$	$O(1)$		√	♯	$1-(1-\rho)^t$
SPDP[4]	MHT	$O(t)$	$O(t)$	$O(t)$	√	√		$1-(1-\rho)^{t\cdot s}$
DPDP-I[5]	MHT	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$		√		$1-(1-\rho)^t$
DPDP-II[5]	MHT	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$				$1-(1-\rho)^{Ω(n)}$
CPOR-I[6]	HomT	$O(t)$	$O(t)$	$O(1)$			♯	$1-(1-\rho)^t$
CPOR-II[6]	HomT	$O(t+s)$	$O(t+s)$	$O(s)$	√		♯	$1-(1-\rho)^{t\cdot s}$
Our Scheme	HomR	$O(t+c\cdot s)$	$O(t+s)$	$O(s)$	√	√	√	$1-\prod_{k\in P}(1-\rho_k)^{r_k\cdot t\cdot s}$

TABLE 1

Comparison of POR/PDP Schemes for a File Consisting of n Blocksin an enterprise may be illegally accessed through a remote interface provided by a multicloud, or relevant data and archives may be lost or tampered with when they are stored into an uncertain storage pool outside the enterprise.Therefore, it is indispensable for cloud service providers (CSPs) to provide security techniques for managing their storage services.Provable data possession (PDP) [2] (or proofs of retrievability (POR) [3]) is such a probabilistic proof technique for a storage provider to prove the integrity and ownership of clients' data without downloading data.The proof-checking without downloading makes it especially important for large-size files and folders (typically including many clients' files) to check whether these data have been tampered with or deleted without downloading the latest version of data. Thus, it is able to replace traditional hash and signature functions in storage outsourcing.Various PDP schemes

have been recently proposed, such as Scalable PDP [4] and Dynamic PDP [5].However, these schemes mainly focus on PDP issues at untrusted servers in a single cloud storage provider and are not suitable for a multicloud environment (see the comparison of POR/PDP schemes in Table 1).

Motivation

To provide a low cost, scalable, location independent platform for managing clients' data, current cloud storage systems adopt several new distributed file systems, for example, Apache Hadoop Distribution File System (HDFS), Google File System (GFS), Amazon S3 File System, CloudStore, etc. These file systems share some similar features: a single metadata server provides centralized management by a global namespace; files are split into blocks or chunks and stored on block servers; and the systems are comprised of interconnected clusters of block servers. Those features enable cloud service providers to store and process large amounts of data. However, it is crucial to offer an efficient verification on the integrity and availability of stored data for detecting faults and automatic recovery. Moreover, this verification is necessary to provide reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures. Although existing schemes can make a false or true decision for data possession without downloading data at untrusted stores, they are not suitable for a distributed cloud storage environment since they were not originally constructed on interactive proof system. For example, the schemes based on Merkle Hash tree (MHT), such as DPDPI, DPDP-II [2], and SPDP [4] in Table 1, use an authenticated skip list to check the integrity of file blocks adjacently in space. Unfortunately, they did not provide any algorithms for constructing distributed Merkle trees that are necessary for efficient verification in a multicloud environment. In addition, when a client asks for a file block, the server needs to send the file block along with a proof for the intactness of the block. However, this process incurs significant communication overhead in a multicloud environment, since the server in one cloud typically needs to generate such a proof with the help of other cloud storage services, where the adjacent blocks are stored. The other schemes, such as PDP [2], CPOR-I, and CPOR-II [6] in Table 1, are constructed on homomorphic verification tags, by which the server can generate tags for multiple file blocks in terms of a single response value. However, that doesn't mean the responses from multiple clouds can be also combined into a single value on the client side. For lack of homomorphic responses, clients must invoke the PDP protocol repeatedly to check the integrity of file blocks stored in multiple cloud servers. Also, clients need to know the exact position of each file block in a multicloud environment. In addition, the verification process in such a case will lead to high communication overheads and computation costs at client sides as well.Therefore, it is of utmost necessary to design a cooperative PDP model to reduce the storage and network overheads and enhance the transparency of verification activities in clusterbased cloud storage systems. Moreover, such a cooperative PDP scheme should provide features for timely detecting abnormality and renewing multiple copies of data.Even though existing PDP schemes have addressed various security properties, such as public verifiability [2],dynamics [5], scalability [4], and privacy preservation [7],we still need a careful consideration of some potential attacks, including two major categories: Data Leakage Attack by which an adversary can easily obtain the stored data through verification process after running or wiretapping sufficient verification communications (see Attacks 1 and 3 in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.66), and Tag Forgery Attack by which a dishonest CSP can deceive the clients (see Attacks 2 and 4 in Appendix A, available in the online supplemental material). These two attacks may cause potential risks for privacy leakage and ownership cheating. Also, these attacks can more easily compromise the security of a distributed cloud system than that of a single cloud system.Although various security models have been proposed for existing PDP schemes [2], [7], [6], these models still cannot cover all security requirements, especially for provable secure privacy preservation and ownership authentication. To establish a highly effective security model, it is necessary to analyze the PDP scheme within the framework of zero-knowledge proof system (ZKPS) due to the reason that PDP system is essentially an interactive proof system (IPS), which has been well studied in the cryptography community. In summary, a verification scheme for data integrity in distributed storage environments should have the following features:

- Usability aspect. A client should utilize the integrity check in the way of collaboration services. The scheme should conceal the details of the storage to reduce the burden on clients;
- Security aspect. The scheme should provide adequate security features to resist some existing attacks, such as data leakage attack and tag forgery attack;

- Performance aspect. The scheme should have the lower communication and computation overheads than noncooperative solution.

Related works

To check the availability and integrity of outsourced data in cloud storages, researchers have proposed two basic approaches called PDP [2] and POR [3]. Ateniese et al. [2] first proposed the PDP model for ensuring possession of files on untrusted storages and provided an RSA-based scheme for a static case that achieves the $O(1)$ communication cost. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the server for data possession. This property greatly extended application areas of PDP protocol due to the separation of data owners and the users. However, these schemes are insecure against replay attacks in dynamic scenarios because of the dependencies on the index of blocks. Moreover, they do not fit for multicloud storage due to the loss of homomorphism property in the verification process.In order to support dynamic data operations, Ateniese et al. developed a dynamic PDP solution called Scalable PDP [4]. They proposed a lightweight PDP scheme based on cryptographic hash function and symmetric key encryption, but the servers can deceive the owners by using previous metadata or responses due to the lack of randomness in the challenges. The numbers of updates and challenges are limited and fixed in advance and users cannot perform block insertions anywhere. Based on this work, Erway et al. [5] introduced two Dynamic PDP schemes with a hash function tree to realize $O(\log n)$ communication and computational costs for a n-block file. The basic scheme, called DPDP-I, retains the drawback of Scalable PDP, and in the "blockless" scheme, called DPDPII,the data blocks $\{m_{i_j}\}_{j\in[1,t]}$ can be leaked by the response of a challenge, $M = \sum_{j=1}^{t} a_j m_{i_j}$, where aj is a random challenge value. Furthermore, these schemes are also not effective for a multicloud environment because the verification path of the challenge block cannot be stored completely in a cloud [8]. Juels and Kaliski [3] presented a POR scheme, which relies largely on preprocessing steps that the client conductsBefore sending a file to a CSP. Unfortunately, these operations prevent any efficient extension for updating data. Shacham and Waters [6] 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 cost for t challenge blocks, but their solution is also static and could not prevent the leakage of data blocks in the verification process. Wang et al. [7] presented a dynamic scheme with $O(\log n)$ cost by integrating the Compact POR scheme and MHT into the DPDP. Furthermore, several POR schemesand models have been recently proposed including [9], [10].In [9], Bowers et al. introduced a distributed cryptographic system that allows a set of servers to solve the PDP problem. This system is based on an integrity-protected error correcting code (IP-ECC), which improves the security and efficiency of existing tools, like POR. However, a file must be transformed into l distinct segments with the same length, which are distributed across l servers. Hence, this system is more suitable for RAID rather than cloud storage. Our contributions. In this paper, we address theProblem of provable data possession in distributed cloud environments from the following aspects: high security, transparent verification, and high performance. To achieve these goals, we first propose a verification framework for multicloud storage along with two fundamental techniques: hash index hierarchy (HIH) and homomorphic verifiable response (HVR).We then demonstrate that the possibility of constructing a cooperative PDP (CPDP) scheme without compromising data privacy based on modern cryptographic techniques, such as interactive proof system. We further introduce an effective construction of CPDP scheme using above-mentioned structure. Moreover, we give a security analysis of our CPDP scheme from the IPS model. We prove that this construction is a multiprover zero-knowledge proof system (MP-ZKPS) [11], which has completeness, knowledge soundness, and zero-knowledge properties. These properties ensure that CPDP scheme can implement the security against data leakage attack and tag forgery attack. To improve the system performance with respect to our scheme, we analyze the performance of probabilistic queries for detecting abnormal situations. This probabilistic method also has an inherent benefit in reducing computation and communication overheads. Then, we present an efficient method for the selection of optimal parameter values to minimize the computation overheads of CSPs and the clients' operations. In addition, we analyze that our scheme is suitable for existing distributed cloud storage systems. Finally, our experiments show that our solution introduces very limited computation and communication overheads.

Organization.

The rest of this paper is organized as follows: in Section 2, we describe a formal definition of CPDP and the underlying techniques, which are utilized in the

construction of our scheme. We introduce the details of cooperative PDP scheme for multicloud storage in Section 3.We describe the security and performance evaluation of our scheme in Sections 4 and 5, respectively. We discuss the related work in Sections 1 and 6 conclude this paper.

2. STRUCTURE AND TECHNIQUES

In this section, we present our verification framework for multicloud storage and a formal definition of CPDP. We introduce two fundamental techniques for constructing our CPDP scheme: hash index hierarchy on which the responses of the clients' challenges computed from multiple CSPs can be combined into a single response as the final result; and homomorphic verifiable response which supports distributed cloud storage in a multicloud storage and implements an efficient construction of collision-resistant hash function, which can be viewed as a random oracle model in the verification protocol.

2.1 Verification Framework for Multicloud

Although existing PDP schemes offer a

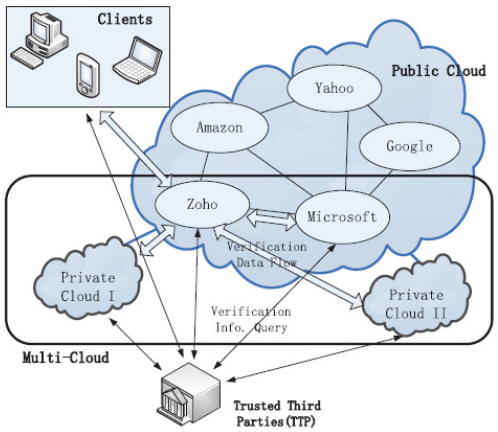

Fig. 1. Verification architecture for data integrity.

publicly accessible remote interface for checking and managing the tremendous amount of data, the majority of existing PDP schemes is incapable to satisfy the inherent requirements from multiple clouds in terms of communication and computation costs. To address this problem, we consider a multicloud storage service as illustrated in Fig. 1. In this architecture, a data storage service involves three different entities: clients who have a large amount of data to be stored in multiple clouds and have the permissions to access and manipulate storeddata; cloud service providers (CSPs) who work together to provide data storage services and have enough storages and computation resources; and Trusted Third Party (TTP) who is trusted to store verification parameters and offer public query services for these parameters. In this architecture, we consider the existence of multiple CSPs to cooperatively store and maintain the clients' data. Moreover, a cooperative PDP is used to verify the integrity and availability of their stored data in all CSPs. to generate and store data owner's public key; and to store the public parameters used to execute the verification protocol in the CPDP scheme. Note that the TTP is not directly involved in the CPDP scheme in order to reduce the complexity of cryptosystem.

2.2 Definition of Cooperative PDP

In order to prove the integrity of data stored in a multicloud environment, we define a framework for CPDP based on IPS and multiprover zero-knowledge proof system (MPZKPS), as follows.

Definition 1 (Cooperative-PDP):

A cooperative provable data possession $S = (KeyGen, TagGen, Proof)$ is a collection of two algorithms (KeyGen, TagGen) and an interactive proof system Proof, as follows:

- $KeyGen(1^{\kappa})$: *takes a security parameter* κ *as input, and returns a secret key* sk *or a public-secret keypair* (pk, sk);
- $TagGen(sk, F, \mathcal{P})$: *takes as inputs a secret key* sk, *a file* F, *and a set of cloud storage providers* $\mathcal{P} = \{P_k\}$, *and returns the triples* (ζ, ψ, σ), *where* ζ *is the secret in tags,* $\psi = (u, \mathcal{H})$ *is a set of verification parameters* u *and an index hierarchy* $\mathcal{H}$ *for* F, $\sigma = \{\sigma^{(k)}\}_{P_k \in \mathcal{P}}$ *denotes a set of all tags,* $\sigma^{(k)}$ *is the tag of the fraction* $F^{(k)}$ *of* F *in* P_k;
- $Proof(\mathcal{P}, V)$: *is a protocol of proof of data possession between CSPs* $(\mathcal{P} = \{P_k\})$ *and a verifier* (V), *that is,*

A trivial way to realize the CPDP is to check the data stored in each cloud one by one, i.e., A trivial way to realize the CPDP is to check the data stored in each cloud one by one, i.e.,

$$\bigwedge_{P_k \in \mathcal{P}} \langle P_k(F^{(k)}, \sigma^{(k)}) \longleftrightarrow V \rangle (pk, \psi),$$

Where $\bigwedge$ denotes the logical AND operations among the boolean outputs of all protocols $\langle P_k, V \rangle$ for all $P_k \in \mathcal{P}$. .However, it would cause significant communication and computation overheads for the verifier, as well as a loss of location-transparent. Such a primitive approach obviously diminishes the advantages of cloud storage: scaling arbitrarily up and down on-demand [13]. To solve this problem, we extend above definition by adding an organizer (O), which is one of CSPs that directly contactsWith the verifier, as follows:

$$\left\langle \sum_{P_k \in \mathcal{P}} P_k(F^{(k)}, \sigma^{(k)}) \longleftrightarrow O \longleftrightarrow V \right\rangle (pk, \psi),$$

Where the action of organizer is to initiate and organize the verification process. This definition is consistent with aforementioned architecture, e.g., a client (or an authorizedApplication) is considered as V , the CSPs are as $\mathcal{P} = \{P_i\}_{i \in [1,c]}$, and the Zoho cloud is as the organizer in Fig. 1. Often, the organizer is an independent server or a certain CSP in P. The advantage of this new multiprover proof system is that it does not make any difference for the clients between multiprover verification process andSingle-prover verification process in the way of collaboration. Also, this kind of transparent verification is able to conceal the details of data storage to reduce the burden on clients. For the sake of clarity, we list some used signals in Table 2.

2.3 Hash Index Hierarchy for CPDP

To support distributed cloud storage, we illustrate a representative architecture used in our cooperative PDP scheme as shown in Fig. 2. Our architecture has a hierarchy structure which resembles a natural representation of file storage. This hierarchical structure H consists of three layers to represent relationships among all blocks for stored resources. They are described as follows:
1. Express layer. Offers an abstract representation of the stored resources;
2. Service layer. Offers and manages cloud storage services; and
3. Storage layer. Realizes data storage on many physical devices. In turn, each CSP fragments and stores the assigned data into the storage servers in Storage Layer. We also make use of colors to distinguish different CSPs. Moreover, we follow the logical order of the data blocks to organize the Storage Layer. The fragment structure is a data structure that maintains a set of block-tag pairs, allowing searches, checks, and updates in $O(1)$ time. An instance ofthis structure is shown in storage layer of Fig. 2: an outsourced file F is split into n blocks $\{m_1, m_2, \ldots, m_n\}$, and each block mi is split into s sectors $\{m_{i,1}, m_{i,2}, \ldots, m_{i,s}\}$. the fragment structure consists of n block-tag pair (m_i, σ_i), ,where is a signature tag of block mi generated by a set of secrets $\tau = (\tau_1, \tau_2, \ldots, \tau_s)$. . In order to check the data integrity, the fragment structure implements probabilistic verification as follows: given a random chosen challenge (or query $Q = \{(i, v_i)\}_{i \in_R I}$, where I am a subset of the block indices and VI is a random coefficient. There exists an efficient algorithm to produce a constant-size response $(\mu_1, \mu_2, \ldots, \mu_s, \sigma')$, Where come from Given a collision-resistant hash function we make use of this architecture to construct a Hash Index Hierarchy H (viewed as a random oracle), which is used to replace the Common hash functions in prior PDP schemes, as follows:

1. **Express layer.** Given s random $\{\tau_i\}_{i=1}^{s}$ and the file name F_n, sets

$$\xi^{(1)} = H_{\sum_{i=1}^{s} \tau_i}(F_n)$$

and makes it public for verification but makes $\{\tau_i\}_{i=1}^{s}$ secret.

2. **Service layer.** Given the $\xi^{(1)}$ and the cloud name C_k, sets $\xi_k^{(2)} = H_{\xi^{(1)}}(C_k)$.

3. **Storage layer.** Given the $\xi^{(2)}$, a block number i, and its index record $\chi_i = "B_i \| V_i \| R_i"$, sets $\xi_{i,k}^{(3)} = H_{\xi^{(2)}}(\chi_i)$, where B_i is the sequence number of a block, V_i^{jk} is the updated version number, and R_i is a random integer to avoid collision.

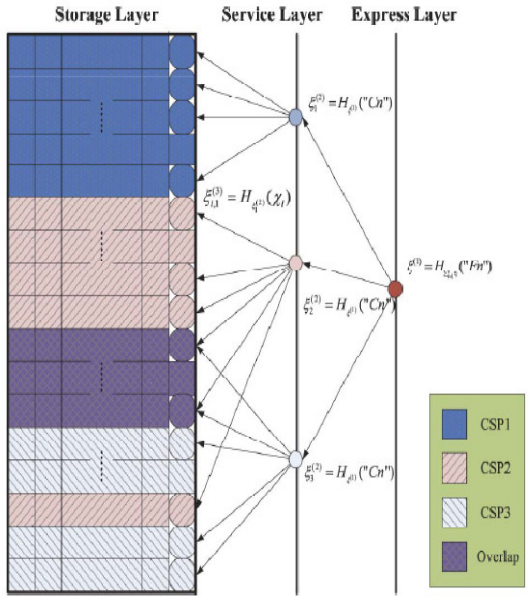

Fig. 2. Index hash hierarchy of CPDP model.

As a virtualization approach, we introduce a simple index-hash table to record the changes of file blocks as well as to generate the hash value of each block in the verification process. The structure of is similar to the structure of file block allocation table in file systems. The index-hash table consists of serial number, block number,version number, random integer, and so on. Different from the common index table, we assure that all records in our index table differ from one another to prevent forgery of data blocks and tags. By using this structure, especially the index records fig, our CPDP scheme can also supportDynamic data operations [8].The proposed structure can be readily incorperated intoMAC-based, ECC, or RSA schemes [2], [6]. These schemes, built from collision-resistance signatures (see Section 3.1)
and the random oracle model, have the shortest query andresponse with public verifiability. They share several common characters for the implementation of the CPDP framework in the multiple clouds:
1. A file is split into n _ s sectors and each block (s sectors) corresponds to a tag, so that the storage of signature tags can be reduced by the increase of s;
2. A verifier can verify the integrity of file in random sampling approach, which is of utmost importance for large files;
3. These schemes rely on homomorphism properties to aggregate data and tags into a constant-size response, which minimizes the overhead of network communication; and
4. The hierarchy structure provides a virtualization approach to conceal the storage details of multiple CSPs.

2.4 Homomorphic Verifiable Response for CPDP

A homomorphism is a map $f : \mathbb{IP} \longrightarrow \mathbb{QQ}$ between two groups such that $f(g_1 \oplus g_2) = f(g_1) \otimes f(g_2)$ for all g1;
where denotes the operation in IP and denotes the operation in QQ. This notation has been used to define Homomorphic Verifiable Tags (HVTs) in [2]: given two values _i and _j for two messages mi and mj, anyone can combine them into a value _0 corresponding to the sum of the messages mi þ mj. When provable data possession is considered as a challenge-response protocol,we extend this notation to the concept of HVR, which is used to integrate multiple responses from the different CSPs in CPDP scheme as follows.

Definition 2 (Homomorphic Verifiable Response). *A response is called homomorphic verifiable response in a PDP protocol, if given two responses θ_i and θ_j for two challenges Q_i and Q_j from two CSPs, there exists an efficient algorithm to combine them into a response θ corresponding to the sum of the challenges $Q_i \bigcup Q_j$.*

Homomorphic verifiable response is the key technique of CPDP because it not only reduces the communication bandwidth, but also conceals the location of outsourced data in the distributed cloud storage environment

3 COOPERATIVE PDP SCHEME

In this section, we propose a CPDP scheme for multicloud system based on the above-mentioned structure and techniques.This scheme is constructed on collision-resistant hash, bilinear map group, aggregation algorithm, and homomorphic responses.

3.1 Notations and Preliminaries

Let $\mathbb{H} = \{H_k\}$ be a family of hash functions $H_k : \{0,1\}^n \rightarrow \{0,1\}^s$ index by $k \in K$. We say that algorithm A has advantage ϵ in breaking collision-resistance of $\mathbb{H}$ if

$$\Pr[A(k) = (m_0, m_1) : m_0 \neq m_1, H_k(m_0) = H_k(m_1)] \geq \epsilon,$$

Where the probability is over the random choices of $k \in \mathcal{K}$ and the random bits of A. So that, we have the following definition.

Definition 3 (Collision-Resistant Hash). *A hash family $\mathbb{H}$ is (t,ϵ)-collision-resistant if no t-time adversary has advantage at least ϵ in breaking collision-resistance of $\mathbb{H}$.*

Definition 4 (Bilinear Map Group System). *A bilinear map group system is a tuple $\mathbb{S} = \langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ composed of the objects as described above.*

3.2 Our CPDP Scheme

In our scheme (see Fig. 3), the manager first runs algorithm KeyGen to obtain the public/private key pairs for CSPs andUsers. Then, the clients generate the tags of outsourced data by using TagGen. Anytime, the protocol Proof is performedby a five-move interactive proof protocol between a verifier and more than one CSP, in which CSPs need not to interact with each other during the verification process, but anOrganizer is used to organize and manage all CSPs.This protocol can be described as follows:

1. The organizer initiates the protocol and sends a commitment to the verifier;
2. The verifier returns a challenge set of random index coefficient pairs Q to the organizer;
3. The organizer relays them into each Pi in P according to the exact position of each data block;
4. Each Pi returns its response of challenge to the organizer; and
5. The organizer synthesizes a final response from received responses and sends it to the verifier. The above process would guarantee that the verifier accesses files without knowing on which CSPs or in what geographical locations their files reside.

4. SECURITY ANALYSES

We give a brief security analysis of our CPDP construction. This construction is directly derived from multiprover zero knowledge proof system (MP-ZKPS), which satisfies following properties for a given assertion L:

1. Completeness. Whenever $x \in L$, here exists a strategy for the provers that convince the verifier that this is the case.
2. Soundness. Whenever $x \notin L$, whatever strategy the provers employ, they will not convince the verifier that $x \in L$.
3. Zero knowledge. No cheating verifier can learn anything other than the veracity of the statement.

According to existing IPS research [15], these properties can protect our construction from various attacks, such as data leakage attack (privacy leakage), tag forgery attack (ownership cheating), etc. In details, the security of our scheme can be analyzed as follows.

4.1 Collision Resistant for Index Hash Hierarchy

In our CPDP scheme, the collision resistant of index hash hierarchy is the basis and prerequisite for the security of whole scheme, which is described as being secure in the random oracle model. Although the hash function is collision resistant, a successful hash collision can still be used toproduce a forged tag when the same hash value is reused multiple times, e.g., a legitimate client modifies the data or repeats to insert and delete data blocks of outsourced data.To avoid the hash collision, the hash value $\xi_{i,k}^{(3)}$, which is sed to generate the tag i in CPDP scheme, is computed from the set of values $\{\tau_i\}, F_n, C_k, \{\chi_i\}$; fig. As long as there exists 1 bit difference in these data, we can avoid the hash collision. As a consequence, we have the following theorem(see Appendix B, available in the online supplemental material).

Theorem 1 (Collision Resistant). The index hash hierarchy in CPDP scheme is collision resistant, even if the client generates

$$\sqrt{2p \cdot \ln \frac{1}{1-\varepsilon}}$$

Files with the same file name and cloud name, and the client repeats

$$\sqrt{2^{L+1} \cdot \ln \frac{1}{1-\varepsilon}}$$

Times to modify, insert, and delete data blocks, where the collision probability is at least ε, $\tau_i \in \mathbb{Z}_p$, and $|R_i| = L$ *for* $R_i \in \chi_i$.

4.2 Completeness Property of Verification

In our scheme, the completeness property implies public verifiability property, which allows anyone, not just the client (data owner), to challenge the cloud server for data integrity and data ownership without the need for any

secret information. First, for every available data-tag pair $(F, \sigma) \in$ and a random challenge $Q = (i, v_i)_{i \in I'}$, the verification protocol should be completed with success probability according to the (3), that is,

$$\Pr\left[\left\langle \sum_{P_k \in \mathcal{P}} P_k(F^{(k)}, \sigma^{(k)}) \leftrightarrow O \leftrightarrow V \right\rangle (pk, \psi) = 1\right] = 1.$$

In this process, anyone can obtain the owner's public key $pk = (g, h, H_1 = h^{\alpha}, H_2 = h^{\beta})$ and the corresponding file parameter from TTP to execute the verification protocol, hence this is a public verifiable protocol. Moreover, for different owners, the secrets and hidden in their public key pk are also different, determining that a success verification can only be implemented by the real owner's public key. In addition, the parameter is used to store the file-related information, so an owner can employ a unique public key to deal with a large number of outsourced files.

4.3 Zero-Knowledge Property of Verification

The CPDP construction is in essence a Multi-Proven Zero knowledge Proof (MP-ZKP) system [11], which can be considered as an extension of the notion of an IPS. Roughly speaking, in the scenario of MP-ZKP, a polynomial-time bounded verifier interacts with several provers whose computational powers are unlimited. According to a Simulator model, in which every cheating verifier has a simulator that can produce a transcript that "looks like" and Interaction between a honest prover and a cheating verifier, we can prove our CPDP construction has Zero-knowledge property (see Appendix C, available in the online supplemental material)

$$\begin{aligned}
\pi' \cdot e(\sigma', h) &= \prod_{j=1}^{s} e\left(u_j^{\lambda_j}, H_2'\right) \cdot e\left(\prod_{(i,v_i)\in Q} \sigma_i^{v_i \cdot \gamma}, h\right) \\
&= \prod_{j=1}^{s} e\left(u_j^{\lambda_j}, H_2'\right) \cdot e\left(\prod_{(i,v_i)\in Q} \left((\xi_{i,k}^{(3)})^{\alpha} \cdot \left(\prod_{j=1}^{s} u_j^{m_{ij}}\right)^{\beta}\right)^{v_i \cdot \gamma}, h\right) \\
&= \prod_{j=1}^{s} e\left(u_j^{\gamma \cdot \lambda_j}, H_2\right) \cdot e\left(\prod_{(i,v_i)\in Q} (\xi_i^{(3)})^{v_i}, h\right)^{\alpha\gamma} \\
&\quad \cdot e\left(\prod_{j=1}^{s} u_j^{\sum_{(i,v_i)\in Q} \gamma m_{ij} v_i}, h^{\beta}\right) \\
&\quad - e\left(\prod_{(i,v_i)\in Q} (\xi_i^{(3)})^{v_i}, H_1'\right) \cdot \prod_{j=1}^{s} e\left(u_j^{\mu_j'}, H_2\right).
\end{aligned}$$

Theorem 2 (Zero-Knowledge Property). *The verification protocol Proof(P, V) in CPDP scheme is a computational zero-knowledge system under a simulator model, that is, for every probabilistic polynomial-time interactive machine V*, there exists a probabilistic polynomial-time algorithm S* such that the ensembles* $View\left(\sum_{P_k \in \mathcal{P}} P_k(F^{(k)}, \sigma^{(k)}) \leftrightarrow O \leftrightarrow V^*\right)(pk, \psi))$ *and* $S^*(pk, \psi)$ *are computationally indistinguishable.*

Zero-knowledge is a property that achieves the CSPs' robustness against attempts to gain knowledge by interacting with them. For our construction, we make use of the zero-knowledge property to preserve the privacy of data blocks and signature tags. First, randomness is adopted into the CSPs' responses in order to resist the data leakage attacks (see Attacks 1 and 3 in Appendix A, available in the online supplemental material). That is, the random integer $\lambda_{j,k}$ is introduced into the response $\mu_{j,k}$, i.e., $\mu_{j,k} = \lambda_{j,k} + \sum_{(i,v_i)\in Q_k} v_i \cdot m_{i,j}$. This means that the cheating verifier cannot obtain $m_{i,j}$ from $\mu_{j,k}$ because he does not know the random integer $\lambda_{j,k}$. at the same time, a random integer _is also introduced to randomize the verification tag i.e. thus, the tag _ cannot reveal to the cheating verifier in terms of randomness.

$$\sigma' \leftarrow \left(\prod_{P_k \in \mathcal{P}} \sigma_k' \cdot R_k^{-s}\right)^{\gamma}.$$

TABLE 4
Comparison of Communication Overheads between Our CPDP and Noncooperative (Trivial) Scheme

	CPDP Scheme	Trivial Scheme
Commitment	l_2	cl_2
Challenge1	$2tl_0$	$2tl_0$
Challenge2	$2tl_0/c$	
Response1	$sl_0 + 2l_1 + l_T$	$(sl_0 + l_1 + l_T)c$
Response2	$sl_0 + l_1 + l_T$	

5 PERFORMANCE EVALUATIONS

In this section, to detect abnormality in a low overhead and timely manner, we analyze and optimize the performance of CPDP scheme based on the above scheme from two aspects: evaluation of probabilistic queries and optimization of length of blocks. To validate the effects of scheme, we introduce a prototype of CPDP-based audit system and present the experimental results.

5.1 Performance Analysis for CPDP Scheme

We present the computation cost of our CPDP scheme in Table 3. We use $[E]$ to denote the computation cost of an exponent operation in $\mathbb{G}$, namely, g^x where x is a positive integer in $\mathbb{Z}_p$ and $g \in \mathbb{G}$ or $\mathbb{G}_T$. . We neglect the computation cost of algebraic operations and simple modular arithmetic operations because they run fast enough [16]. The most complex operation is the computation of a bilinear map $e(\cdot,\cdot)$ between two elliptic points (denoted as $[B]$).Then, we analyze the storage and communication costs of our scheme. We define the bilinear pairing takes the form $e : E(\mathbb{F}_{p^m}) \times E(\mathbb{F}_{p^{km}}) \rightarrow \mathbb{F}^*_{p^{km}}$ (the definition given here is from [17], [18]), where p is a prime, m is a positive integer,And k is the embedding degree (or security multiplier). In this case, we utilize an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ to replace the symmetric pairing in the original schemes. In Table 3, it is easy to find that client's computation overheads are entirely irrelevant for the number of CSPs. Further, our scheme has better performance compared with no cooperative approach due to the total of computation overheads decrease $3(c-1)$ times bilinear map operations, where c is the number of clouds in a multicloud.

The reason is that, before the responses are sent to the verifier from c clouds, the organizer has aggregate these responses into a response by using aggregation algorithm, so the verifier only need to verify this response once to obtain the final result. Without loss of generality, let the security parameter _ be 80 bits, we need the elliptic curve domain parameters over $|p| = 160$ and $m = 1$ in our experiments. This means that the length of integer is $l_0 = 2\kappa$ in $\mathbb{Z}_p$. similarly, we have $l_1 = 4\kappa$ in $\mathbb{G}_1$, $l_2 = 24\kappa$ in $\mathbb{G}_2$, and $l_T = 24\kappa$ in $\mathbb{G}_T$ for the embedding degree $k = 6$. the storage and communication costs of our scheme is shown in Table 4. The storage overhead of a file with $size(f) = 1$ M-bytes .This is $store(f) = n \cdot s \cdot l_0 + n \cdot l_1 = 1.04$ bytes.Thestorage overhead of its index table is n _ l0 ¼ 20 K-bytes. We define the overhead rate and it should therefore be kept as low as possible in order to minimize the storage in cloud storage providers. It is obvious that a higher s means much lower storage. Furthermore, in the verification protocol, the communication overhead of challenge is $2t \cdot l_0 = 40 \cdot$ t-Bytes in terms of the number of challenged blocks t, but its response (Response1 or Response2) has a constant-size communication overhead $s \cdot l_0 + l_1 + l_T \approx$ 1:3 K-bytes for different file sizes. Also, it implies that client's communication overheads are of a fixed size, which is entirely irrelevant for the number of CSPs.

5.2 Probabilistic Verification

We recall the probabilistic verification of common PDP scheme (which only involves one CSP), in which the verification process achieves the detection of CSP server misbehavior in a random sampling mode in order to reduce the workload on the server. The detection probability of disrupted blocks P is an important parameter to guarantee that these blocks can be detected in time. Assume the CSP modifies e blocks out of the n-block file, that is, the probability of disrupted blocks is $\rho_b = \frac{e}{n} \cdot$. Let t be the number of queried blocks for a challenge in the verification protocol. We have detection probability2

$$P(\rho_b, t) \geq 1 - \left(\frac{n-e}{n}\right)^t = 1 - (1-\rho_b)^t,$$

Another advantage of probabilistic verification based on random sampling is that it is easy to identify the tampering orForging data blocks or tags. The identification function is obvious: when the verification fails, we can choose the partialset of challenge indexes as a new challenge set, and continue to execute the verification protocol. The above search process can be repeatedly executed until the bad block is found. The complexity of such a search process is $O(\log n)$.

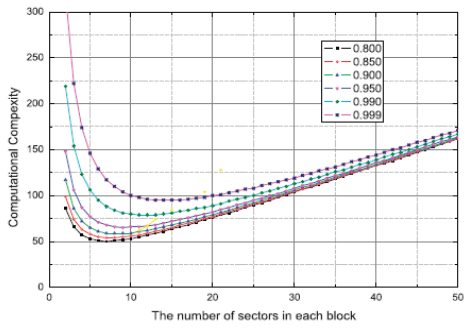

Fig. 4. The relationship between computational cost and the number of sectors in each block.

$$t \approx \frac{\log(1-P)}{s \cdot \sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)},$$

5.3 Parameter Optimization

In the fragment structure, the number of sectors per blocks is an important parameter to affect the performance of storage services and audit services. Hence, we propose an Optimization algorithm for the value of s in this section. Our results show that the optimal value cannot only minimize the computation and communication overheads, but also reduce the size of extra storage, which is required to store the verification tags in CSPs.Assume ρ denotes the probability of sector corruption. In the fragment structure, the choosing of s is extremely important for improving the performance of the CPDP scheme. Given the detection probability P and the probability of sector corruption ρ for multiple clouds $\mathcal{P} = \{P_k\}$,the optimal value of s can be computed by

$$\min_{s \in \mathbb{N}} \left\{ \frac{\log(1-P)}{\sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)} \cdot \frac{a}{s} + b \cdot s + c \right\},$$

Where a.t+b.s+c denote the computational cost of verification protocol in PDP scheme,a,b,c€IR and c is a constant. This conclusion can be obtained from following Process. According to abovementioned results, the sampling probability holds.

$$w \geq \frac{\log(1-P)}{sz \cdot \sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)} = \frac{\log(1-P)}{n \cdot s \cdot \sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)}.$$

In order to minimize the computational cost, we have

$$\begin{aligned} &\min_{s \in \mathbb{N}} \{a \cdot t + b \cdot s + c\} \\ &= \min_{s \in \mathbb{N}} \{a \cdot n \cdot w + b \cdot s + c\} \\ &\geq \min_{s \in \mathbb{N}} \left\{ \frac{\log(1-P)}{\sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)} \frac{a}{s} + b \cdot s + c \right\}, \end{aligned}$$

where r_k denotes the proportion of data blocks in the kth CSP, ρ_k denotes the probability of file corruption in the kth CSP. Since $\frac{a}{s}$ is a monotone decreasing function and b.s is a monotone increasing function for s > 0, there exists an optimal value ofs€n in the above equation. The optimal value of s is unrelated to a certain file from this conclusion if the probability ρ is a constant value. For instance, we assume a multicloud storage involves three CSPs and the probability of sector corruption is a constant value. We set the detection probability P with the range from 0.8 to 1, e.g $P - \{0.8, 0.85, 0.9, 0.95, 0.99, 0.999\}$. . For a file, the proportion of data blocks is 50, 30, and 20 percent in three CSPs, respectively, that is, In terms of Table 3, the computational cost of CSPs can be simplified to t+3s+9. Then, we can observe the computational cost under different s and P in Fig.4

TABLE 5
The Influence of s, t under the Different Corruption Probabilities ρ and the Different Detection Probabilities P

$\mathcal{P}$	{0.1,0.2,0.01}	{0.01,0.02,0.001}	{0.001,0.002,0.0001}	{0.0001,0.0002,0.00001}
r	{0.5,0.3,0.2}	{0.5,0.3,0.2}	{0.5,0.3,0.2}	{0.5,0.3,0.2}
0.8	3/4	7/20	23/62	71/202
0.85	3/5	8/21	26/65	79/214
0.9	3/6	10/20	28/73	87/226
0.95	3/8	11/29	31/86	100/269
0.99	4/10	13/31	39/105	119/345
0.999	5/11	16/38	45/125	140/420

.

TABLE 6
The Influence of Parameters under Different Detection Probabilities P ($\mathcal{P} = \{\rho_1, \rho_2, \rho_3\} = \{0.01, 0.02, 0.001\}$, $\{r_1, r_2, r_3\} = \{0.5, 0.3, 0.2\}$)

P	0.8	0.85	0.9	0.95	0.99	0.999
$sz \cdot w$	142.60	168.09	204.02	265.43	408.04	612.06
s	7	8	10	11	13	16
t	20	21	20	29	31	38

6. CONCLUSIONS

In this paper, we presented the construction of an efficient PDP scheme for distributed cloud storage. Based on homomorphic verifiable response and hash index hierarchy, we have proposed a cooperative PDP scheme to support dynamic scalability on multiple storage servers. We also showed that our scheme provided all security properties required by zero-knowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. Furthermore, we optimized the probabilistic query and periodic verification to improve the audit performance. Our experiments clearly demonstrated that our approaches only introduce a small amount of computation and communication overheads. Therefore, our solution can be treated as a new candidate for data integrity verification in outsourcing data storage systems. As part of future work, we would extend our work to explore more effective CPDP constructions. First, from our experiments we found that the performance of CPDPscheme, especially for large files, is affected by the bilinear mapping operations due to its high complexity. To solve this problem, RSA-based constructions may be a better choice, but this is still a challenging task because the existing RSA-based schemes have too many restrictions on the performance and security [2]. Next, from a practical point of view, we still need to address some issues about integrating our CPDP scheme smoothly with existing systems, for example, how to match index-hash hierarchy with HDFS's two-layer name space, how to match index structure with cluster-network model, and how to dynamically update the CPDP parameters according to HDFS' specific requirements. Finally, it is still a challenging problem for the generation of tags with the length irrelevant to the size of data blocks. We would explore such an issue to provide the support of variable-length block verification.

References

[1] B. Sotomayor, R.S. Montero, I.M. Llorente, and I.T. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," IEEE Internet Computing, vol. 13, no. 5, pp. 14-22, Sept. 2009.

[2] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J.Peterson, and D.X. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, 2007.

[3] A. Juels and B.S.K. Jr., "Pors: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 584-597, 2007.

[4] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf.Security and Privacy in Comm. Networks (SecureComm '08), pp. 1-10,
2008.

[5] C.C. Erway, A. Ku¨ pc¸u¨ , C. Papamanthou, and R. Tamassia,"Dynamic Provable Data Possession," Proc. 16th ACM Conf.Computer and Comm. Security (CCS '09), pp. 213-222, 2009.

[6] H. Shacham and B. Waters, "Compact Proofs of Retrievability,"Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08),pp. 90-107, 2008.

[7] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.

[8] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S.S. Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," Proc. ACM Symp. Applied Computing, pp. 1550-1557, 2011.

[9] K.D. Bowers, A. Juels, and A. Oprea, "Hail: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf.Computer and Comm. Security, pp. 187-198, 2009.

[10] Y. Dodis, S.P. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Sixth