Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage

Yan Zhu, Hongxin Hu, Gail-Joon Ahn, *Senior Member, IEEE*, Mengyang Yu

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 multi-prover 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.

Index Terms—Storage Security, Provable Data Possession, Interactive Protocol, Zero-knowledge, Multiple Cloud, Cooperative

1 Introduction

In recent years, cloud storage service has become a faster profit 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 multi-cloud 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 multi-cloud, 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 in an enterprise may be illegally accessed through a remote interface provided by a multi-cloud, 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 multi-cloud 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

- *A preliminary version of this paper appeared under the title "Efficient Provable Data Possession for Hybrid Clouds" in Proc. of the 17th ACM Conference on Computer and Communications Security (CCS), Chicago, IL, USA, 2010, pp. 881-883.*
- *Y. Zhu is with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China, and the Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing 100871, China. E-mail: {yan.zhu,huzexing}@pku.edu.cn.*
- *H. Hu and G.-J. Ahn are with the Arizona State University, Tempe, Arizona, 85287. E-mail: {hxhu,gahn}@asu.edu.*
- *M. Yang is with the School of Mathematics Science, Peking University, Beijing 100871, China. E-mail: myyu@pku.edu.cn.*

Digital Object Indentifier 10.1109/TPDS.2012.66

TABLE 1
Comparison of POR/PDP schemes for a file consisting of n blocks.

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	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$				$1-(1-\rho)^{\Omega(n)}$
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	$HomR$	$O(t+c\cdot s)$	$O(t+s)$	$O(s)$	✓	✓	✓	$1-\prod_{P_k\in\mathcal{P}}(1-\rho_k)^{r_k\cdot t\cdot s}$

s is the number of sectors in each block, c is the number of CSPs in a multi-cloud, t is the number of sampling blocks, ρ and ρ_k are the probability of block corruption in a cloud server and k-th cloud server in a multi-cloud $\mathcal{P}=\{P_k\}$, respective, ♯ denotes the verification process in a trivial approach, and $MHT, HomT, HomR$ denotes Merkle Hash tree, homomorphic tags, and homomorphic responses, respectively.

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 DPDP-I, 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 multi-cloud 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 multi-cloud 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 multi-cloud 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 cluster-based 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), and *Tag Forgery Attack* by which a dishonest CSP can deceive the clients (see Attacks 2 and 4 in Appendix A). 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 non-cooperative solution.

Related Works. To check the availability and integrity of outsourced data in cloud storages, researchers have proposed two basic approaches called Provable Data Possession (PDP) [2] and Proofs of Retrievability (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 multi-cloud 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 DPDP-II, 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 a_j is a random challenge value. Furthermore, these schemes are also not effective for a multi-cloud 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 conducts before 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 Merkle Hash Tree (MHT) into the DPDP. Furthermore, several POR schemes and 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 a cloud storage.

Our Contributions. In this paper, we address the problem 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 multi-cloud 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 (IPS). 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 multi-prover 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 multi-cloud storage in Section 3. We describes the security and performance evaluation of our scheme in Section 4 and 5, respectively. We discuss the related work in Section and Section 6 concludes this paper.

2 Structure and Techniques

In this section, we present our verification framework for multi-cloud storage and a formal definition of CPDP. We introduce two fundamental techniques for constructing our CPDP scheme: hash index hierarchy (HIH) on which the responses of the clients' challenges computed from multiple CSPs can be com-

bined into a single response as the final result; and homomorphic verifiable response (HVR) which supports distributed cloud storage in a multi-cloud 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 Multi-Cloud

Although existing PDP schemes offer a publicly accessible remote interface for checking and managing the tremendous amount of data, the majority of existing PDP schemes are incapable to satisfy the inherent requirements from multiple clouds in terms of communication and computation costs. To address this problem, we consider a multi-cloud storage service as illustrated in Figure 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 stored data; 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.

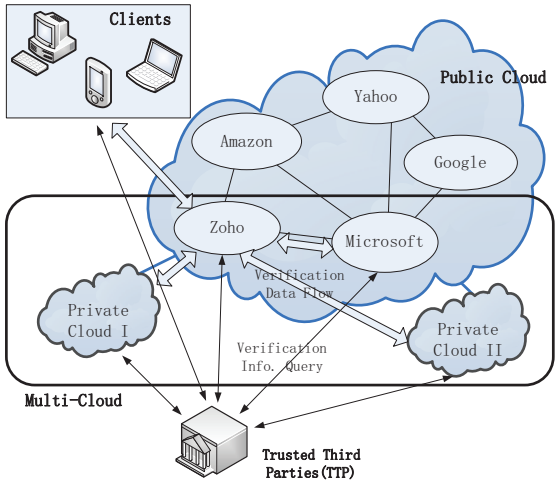


Fig. 1. Verification architecture for data integrity.

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. The verification procedure is described as follows: Firstly, a client (data owner) uses the secret key to pre-process a file which consists of a collection of n blocks, generates a set of public verification information that is stored in TTP, transmits the file and some verification tags to CSPs, and may delete its local copy; Then, by using a verification protocol, the clients can issue a challenge for one CSP to check the integrity and availability of outsourced data with respect to public information stored in TTP.

We neither assume that CSP is trust to guarantee the security of the stored data, nor assume that data owner has the ability to collect the evidence of the CSP's fault after errors have been found. To achieve this goal, a TTP server is constructed as a core trust base on the cloud for the sake of security. We assume the TTP is reliable and independent through the following functions [12]: to setup and maintain the CPDP cryptosystem; 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 multi-cloud environment, we define a framework for CPDP based on interactive proof system (IPS) and multi-prover zero-knowledge proof system (MP-ZKPS), as follows:

Definition 1 (Cooperative-PDP): A cooperative provable data possession $\mathcal{S} = (KeyGen, TagGen, Proof)$ 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 key-pair (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,

$$\left\langle \sum_{P_k \in \mathcal{P}} P_k(F^{(k)}, \sigma^{(k)}) \longleftrightarrow V \right\rangle (pk, \psi) = \begin{cases} 1 & F = \{F^{(k)}\} \text{ is intact} \\ 0 & F = \{F^{(k)}\} \text{ is changed} \end{cases},$$

where each P_k takes as input a file $F^{(k)}$ and a set of tags $\sigma^{(k)}$, and a public key pk and a set of public parameters ψ are the common input between P and V. At the end of the protocol run, V returns a bit $\{0|1\}$ denoting false and true. Where, $\sum_{P_k \in \mathcal{P}}$ denotes cooperative computing in $P_k \in \mathcal{P}$.

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 contacts with 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 authorized application) 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 Figure 1. Often, the organizer is an independent server or a certain CSP in $\mathcal{P}$. The advantage of this new multi-prover proof system is that it does not make any difference for the clients between multi-prover verification process and single-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.

TABLE 2
The signal and its explanation.

Sig.	Repression
n	the number of blocks in a file;
s	the number of sectors in each block;
t	the number of index coefficient pairs in a query;
c	the number of clouds to store a file;
F	the file with $n \times s$ sectors, i.e., $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$;
σ	the set of tags, i.e., $\sigma = \{\sigma_i\}_{i\in[1,n]}$;
Q	the set of index-coefficient pairs, i.e., $Q = \{(i, v_i)\}$;
θ	the response for the challenge Q.

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 Figure 2. Our architecture has a hierarchy structure which resembles a natural representation of file storage. This hierarchical structure $\mathcal{H}$ 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.

We make use of this simple hierarchy to organize data blocks from multiple CSP services into a large-size file by shading their differences among these cloud storage systems. For example, in Figure 2 the resources in Express Layer are split and stored into three CSPs, that are indicated by different colors, in Service Layer. 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. This architecture also provides special functions for data storage and management, e.g., there may exist overlaps among data blocks (as shown in dashed boxes) and discontinuous blocks but these functions may increase the complexity of storage management.

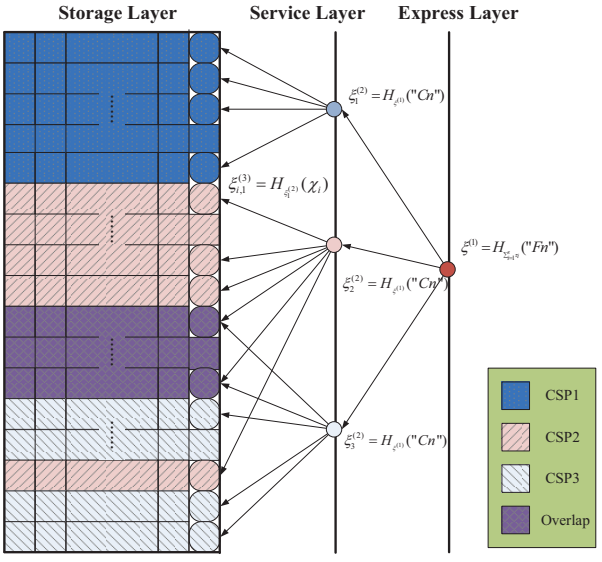


Fig. 2. Index-hash hierarchy of CPDP model.

In storage layer, we define a common fragment structure that provides probabilistic verification of data integrity for outsourced storage. 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 of this structure is shown in storage layer of Figure 2: an outsourced file F is split into n blocks $\{m_1, m_2, \cdots, m_n\}$, and each block m_i is split into s sectors $\{m_{i,1}, m_{i,2}, \cdots, m_{i,s}\}$. The fragment structure consists of n block-tag pair (m_i, σ_i), where σ_i is a signature tag of block m_i generated by a set of secrets $\tau = (\tau_1, \tau_2, \cdots, \tau_s)$. 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 is a subset of the block indices and v_i is a random coefficient. There exists an efficient algorithm to produce a constant-size response $(\mu_1, \mu_2, \cdots, \mu_s, \sigma')$, where μ_i comes from all $\{m_{k,i}, v_k\}_{k\in I}$ and σ' is from all $\{\sigma_k, v_k\}_{k\in I}$.

Given a collision-resistant hash function $H_k(\cdot)$, we make use of this architecture to construct a Hash Index Hierarchy $\mathcal{H}$ (viewed as a random oracle), which is used to replace the common hash function 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 $\xi_{i,k}^{(3)} = H_{\xi_k^{(2)}}(\chi_i)$, where B_i is the sequence number of a block, V_i is the updated version number, and R_i is a random integer to avoid collision.

As a virtualization approach, we introduce a simple index-hash table $\chi = \{\chi_i\}$ 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 χ 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 $\{\chi_i\}$, our CPDP scheme can also support dynamic data operations [8].

The proposed structure can be readily incorperated into MAC-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 and response 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 $n \times s$ 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 homomorphic 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 $f : \mathbb{P} \to \mathbb{Q}$ between two groups such that $f(g_1 \oplus g_2) = f(g_1) \otimes f(g_2)$ for all $g_1, g_2 \in \mathbb{P}$, where $\oplus$ denotes the operation in $\mathbb{P}$ and $\otimes$ denotes the operation in $\mathbb{Q}$. This notation has been used to define Homomorphic Verifiable Tags (HVTs) in [2]: Given two values σ_i and σ_j for two messages m_i and m_j, anyone can combine them into a value σ' corresponding to the sum of the messages $m_i + m_j$. When provable data possession is considered as a challenge-response protocol, we extend this notation to the concept of Homomorphic Verifiable Responses (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 multi-cloud 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 $H_k : \{0,1\}^n \to \{0,1\}^*$ index by $k \in \mathcal{K}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in breaking collision-resistance of $\mathbb{H}$ if $\Pr[\mathcal{A}(k) = (m_0, m_1) : m_0 \neq m_1, H_k(m_0) = H_k(m_1)] \geq \epsilon$, where the probability is over the random choices of $k \in \mathcal{K}$ and the random bits of $\mathcal{A}$. 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 $\mathbb{H}$.

We set up our system using bilinear pairings proposed by Boneh and Franklin [14]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups using elliptic curve conventions with a large prime order p. The function e is a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: for any $G, H \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$, we have 1) Bilinearity: $e([a]G, [b]H) = e(G, H)^{ab}$; 2) Non-degeneracy: $e(G, H) \neq 1$ unless G or $H = 1$; and 3) Computability: $e(G, H)$ is efficiently computable.

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 and users. Then, the clients generate the tags of outsourced data by using $TagGen$. Anytime, the protocol $Proof$ is performed by a 5-move interactive

KeyGen(1^κ): Let $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear map group system with randomly selected generators $g, h \in \mathbb{G}$, where $\mathbb{G}, \mathbb{G}_T$ are two bilinear groups of a large prime order p, $|p| = O(\kappa)$. Makes a hash function $H_k(\cdot)$ public. For a CSP, chooses a random number $s \in_R \mathbb{Z}_p$ and computes $S = g^s \in \mathbb{G}$. Thus, $sk_p = s$ and $pk_p = (g, S)$. For a user, chooses two random numbers $\alpha, \beta \in_R \mathbb{Z}_p$ and sets $sk_u = (\alpha, \beta)$ and $pk_u = (g, h, H_1 = h^\alpha, H_2 = h^\beta)$.

TagGen($sk, F, \mathcal{P}$): Splits F into $n \times s$ sectors $\{m_{i,j}\}_{i\in[1,n],j\in[1,s]} \in \mathbb{Z}_p^{n\times s}$. Chooses s random $\tau_1, \cdots, \tau_s \in \mathbb{Z}_p$ as the secret of this file and computes $u_i = g^{\tau_i} \in \mathbb{G}$ for $i \in [1, s]$. Constructs the index table $\chi = \{\chi_i\}_{i=1}^n$ and fills out the record χ_i [a] in χ for $i \in [1, n]$, then calculates the tag for each block m_i as

$$\begin{cases} \xi^{(1)} \leftarrow H_{\sum_{i=1}^s \tau_i}(F_n), & \xi_k^{(2)} \leftarrow H_{\xi^{(1)}}(C_k), \\ \xi_{i,k}^{(3)} \leftarrow H_{\xi_k^{(2)}}(\chi_i), & \sigma_{i,k} \leftarrow (\xi_{i,k}^{(3)})^\alpha \cdot (\prod_{j=1}^s u_j^{m_{i,j}})^\beta, \end{cases}$$

where F_n is the file name and C_k is the CSP name of $P_k \in \mathcal{P}$. And then stores $\psi = (u, \xi^{(1)}, \chi)$ into TTP, and $\sigma_k = \{\sigma_{i,j}\}_{\forall j=k}$ to $P_k \in \mathcal{P}$, where $u = (u_1, \cdots, u_s)$. Finally, the data owner saves the secret $\zeta = (\tau_1, \cdots, \tau_s)$.

Proof($\mathcal{P}, V$): This is a 5-move protocol among the Provers ($\mathcal{P} = \{P_i\}_{i\in[1,c]}$), an organizer ($O$), and a Verifier ($V$) with the common input (pk, ψ), which is stored in TTP, as follows:

1) **Commitment**($O \to V$): the organizer chooses a random $\gamma \in_R \mathbb{Z}_p$ and sends $H_1' = H_1^\gamma$ to the verifier;
2) **Challenge1**($O \leftarrow V$): the verifier chooses a set of challenge index-coefficient pairs $Q = \{(i, v_i)\}_{i\in I}$ and sends Q to the organizer, where I is a set of random indexes in $[1, n]$ and v_i is a random integer in $\mathbb{Z}_p^*$;
3) **Challenge2**($\mathcal{P} \leftarrow O$): the organizer forwards $Q_k = \{(i, v_i)\}_{m_i\in P_k} \subseteq Q$ to each P_k in $\mathcal{P}$;
4) **Response1**($\mathcal{P} \to O$): P_k chooses a random $r_k \in \mathbb{Z}_p$ and s random $\lambda_{j,k} \in \mathbb{Z}_p$ for $j \in [1, s]$, and calculates a response

$$\sigma_k' \leftarrow S^{r_k} \cdot \prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}, \quad \mu_{j,k} \leftarrow \lambda_{j,k} + \sum_{(i,v_i)\in Q_k} v_i \cdot m_{i,j}, \quad \pi_{j,k} \leftarrow e(u_j^{\lambda_{j,k}}, H_2),$$

where $\mu_k = \{\mu_{j,k}\}_{j\in[1,s]}$ and $\pi_k = \prod_{j=1}^s \pi_{j,k}$. Let $\eta_k \leftarrow g^{r_k} \in \mathbb{G}$, each P_k sends $\theta_k = (\pi_k, \sigma_k', \mu_k, \eta_k)$ to the organizer;
5) **Response2**($O \to V$): After receiving all responses from $\{P_i\}_{i\in[1,c]}$, the organizer aggregates $\{\theta_k\}_{P_k\in\mathcal{P}}$ into a final response θ as

$$\sigma' \leftarrow (\prod_{P_k\in\mathcal{P}} \sigma_k' \cdot \eta_k^{-s})^\gamma, \quad \mu_j' \leftarrow \sum_{P_k\in\mathcal{P}} \gamma \cdot \mu_{j,k}, \quad \pi' \leftarrow (\prod_{P_k\in\mathcal{P}} \pi_k)^\gamma. \tag{1}$$

Let $\mu' = \{\mu_j'\}_{j\in[1,s]}$. The organizer sends $\theta = (\pi', \sigma', \mu')$ to the verifier.

Verification: Now the verifier can check whether the response was correctly formed by checking that

$$\pi' \cdot e(\sigma', h) \stackrel{?}{=} e(\prod_{(i,v_i)\in Q} H_{\xi_k^{(2)}}(\chi_i)^{v_i}, H_1') \cdot e(\prod_{j=1}^s u_j^{\mu_j'}, H_2). \tag{2}$$

a. For $\chi_i =$ "B_i, V_i, R_i" in Section 2.3, we can set $\chi_i = (B_i = i, V_i = 1, R_i \in_R \{0,1\}^*)$ at initial stage of CPDP scheme.

Fig. 3. Cooperative Provable Data Possession for Multi-Cloud Storage.

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 an organizer 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 P_i in $\mathcal{P}$ according to the exact position of each data block; 4) each P_i 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.

In contrast to a single CSP environment, our scheme differs from the common PDP scheme in two aspects:

1) Tag aggregation algorithm: In stage of commitment, the organizer generates a random $\gamma \in_R \mathbb{Z}_p$ and returns its commitment H_1' to the verifier. This assures that the verifier and CSPs do not obtain the value of γ. Therefore, our approach guarantees only the organizer can compute the final σ' by using γ and σ_k' received from CSPs.

After σ' is computed, we need to transfer it to the organizer in stage of "Response1". In order to ensure the security of transmission of data tags, our scheme employs a new method, similar to the ElGamal encryption, to encrypt the combination of tags $\prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}$, that is, for $sk = s \in \mathbb{Z}_p$ and $pk = (g, S = g^s) \in \mathbb{G}^2$, the cipher of message m is $\mathcal{C} = (\mathcal{C}_1 = g^r, \mathcal{C}_2 = m \cdot S^r)$ and its decryption is performed by $m = \mathcal{C}_2 \cdot \mathcal{C}_1^{-s}$. Thus, we hold the equation

$$\begin{aligned} \sigma' &= \left(\prod_{P_k\in\mathcal{P}} \frac{\sigma_k'}{\eta_k^s}\right)^\gamma = \left(\prod_{P_k\in\mathcal{P}} \frac{S^{r_k} \cdot \prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}}{\eta_k^s}\right)^\gamma \\ &= \left(\prod_{P_k\in\mathcal{P}} \cdot \prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}\right)^\gamma = \prod_{(i,v_i)\in Q} \sigma_i^{v_i\cdot\gamma}. \end{aligned}$$

2) Homomorphic responses: Because of the homomorphic property, the responses computed from CSPs

in a multi-cloud can be combined into a single final response as follows: given a set of $\theta_k = (\pi_k, \sigma'_k, \mu_k, \eta_k)$ received from P_k, let $\lambda_j = \sum_{P_k \in \mathcal{P}} \lambda_{j,k}$, the organizer can compute

$$\begin{aligned}\mu'_j &= \sum_{P_k \in \mathcal{P}} \gamma \cdot \mu_{j,k} = \sum_{P_k \in \mathcal{P}} \gamma \cdot \left(\lambda_{j,k} + \sum_{(i,v_i) \in Q_k} v_i \cdot m_{i,j} \right) \\ &= \sum\nolimits_{P_k \in \mathcal{P}} \gamma \cdot \lambda_{j,k} + \gamma \cdot \sum\nolimits_{P_k \in \mathcal{P}} \sum\nolimits_{(i,v_i) \in Q_k} v_i \cdot m_{i,j} \\ &= \gamma \cdot \sum\nolimits_{P_k \in \mathcal{P}} \lambda_{j,k} + \gamma \cdot \sum\nolimits_{(i,v_i) \in Q} v_i \cdot m_{i,j} \\ &= \gamma \cdot \lambda_j + \gamma \cdot \sum\nolimits_{(i,v_i) \in Q} v_i \cdot m_{i,j}.\end{aligned}$$

The commitment of λ_j is also computed by

$$\begin{aligned}\pi' &= (\prod\nolimits_{P_k \in \mathcal{P}} \pi_k)^\gamma = (\prod\nolimits_{P_k \in \mathcal{P}} \prod\nolimits_{j=1}^{s} \pi_{j,k})^\gamma \\ &= \prod\nolimits_{j=1}^{s} \prod\nolimits_{P_k \in \mathcal{P}} e(u_j^{\lambda_{j,k}}, H_2)^\gamma \\ &= \prod\nolimits_{j=1}^{s} e(u_j^{\sum_{P_k \in \mathcal{P}} \lambda_{j,k}}, H_2^\gamma) = \prod\nolimits_{j=1}^{s} e(u_j^{\lambda_j}, H'_2).\end{aligned}$$

It is obvious that the final response θ received by the verifiers from multiple CSPs is same as that in one simple CSP. This means that our CPDP scheme is able to provide a transparent verification for the verifiers. Two response algorithms, Response1 and Response2, comprise an HVR: Given two responses θ_i and θ_j for two challenges Q_i and Q_j from two CSPs, i.e., $\theta_i = Response1(Q_i, \{m_k\}_{k \in I_i}, \{\sigma_k\}_{k \in I_i})$, there exists an efficient algorithm to combine them into a final response θ corresponding to the sum of the challenges $Q_i \bigcup Q_j$, that is,

$$\begin{aligned}\theta &= Response1\left(Q_i \bigcup Q_j, \{m_k\}_{k \in I_i \cup I_j}, \{\sigma_k\}_{k \in I_i \cup I_j}\right) \\ &= Response2(\theta_i, \theta_j).\end{aligned}$$

For multiple CSPs, the above equation can be extended to $\theta = Response2(\{\theta_k\}_{P_k \in \mathcal{P}})$. More importantly, the HVR is a pair of values $\theta = (\pi, \sigma, \mu)$, which has a constant-size even for different challenges.

4 Security Analysis

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

1) **Completeness**: whenever $x \in L$, there exists a strategy for the provers that convinces 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 to produce 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 used to generate the tag σ_i in CPDP scheme, is computed from the set of values $\{\tau_i\}, F_n, C_k, \{\chi_i\}$. As long as there exists one bit difference in these data, we can avoid the hash collision. As a consequence, we have the following theorem (see Appendix B):

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 $(F, \sigma) \in TagGen(sk, F)$ and a random challenge $Q = (i, v_i)_{i \in I}$, the verification protocol should be completed with success probability according to the Equation (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 $\psi = (u, \xi^{(1)}, \chi)$ from TTP to execute the verification protocol, hence this is a public verifiable protocol. Moreover, for different owners, the secrets α and β 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 ψ 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-Prover Zero-knowledge Proof (MP-ZKP) system [11], which can be considered as an extension of the notion of

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

an interactive proof system (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" an interaction between a honest prover and a cheating verifier, we can prove our CPDP construction has Zero-knowledge property (see Appendix C):

Theorem 2 (Zero-Knowledge Property): The verification protocol $Proof(\mathcal{P}, V)$ 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 $View(\langle \sum_{P_k \in \mathcal{P}} P_k(F^{(k)}, \sigma^{(k)}) \leftrightarrow O \leftrightarrow V^* \rangle (pk, \psi))$ 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. Firstly, randomness is adopted into the CSPs' responses in order to resist the *data leakage attacks* (see Attacks 1 and 3 in Appendix A). 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 γ is also introduced to randomize the verification tag σ, i.e., $\sigma' \leftarrow (\prod_{P_k \in \mathcal{P}} \sigma_k' \cdot R_k^{-s})^{\gamma}$. Thus, the tag σ cannot reveal to the cheating verifier in terms of randomness.

4.4 Knowledge soundness of verification

For every data-tag pairs $(F^*, \sigma^*) \notin TagGen(sk, F)$, in order to prove nonexistence of fraudulent $\mathcal{P}^*$ and O^*, we require that the scheme satisfies the knowledge soundness property, that is,

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

where ϵ is a negligible error. We prove that our scheme has the knowledge soundness property by using reduction to absurdity [1]: we make use of $\mathcal{P}^*$ to construct a knowledge extractor $\mathcal{M}$ [7,13], which gets the common input (pk, ψ) and rewindable black-box accesses to the prover P^*, and then attempts to break the computational Diffie-Hellman (CDH) problem in $\mathbb{G}$: given $G, G_1 = G^a, G_2 = G^b \in_R \mathbb{G}$, output $G^{ab} \in \mathbb{G}$. But it is unacceptable because the CDH problem is widely regarded as an unsolved problem in polynomial-time. Thus, the opposite direction of the theorem also follows. We have the following theorem (see Appendix D):

Theorem 3 (Knowledge Soundness Property): Our scheme has (t, ϵ') knowledge soundness in random oracle and rewindable knowledge extractor model assuming the (t, ϵ)-computational Diffie-Hellman (CDH) assumption holds in the group $\mathbb{G}$ for $\epsilon' \geq \epsilon$.

Essentially, the soundness means that it is infeasible to fool the verifier to accept false statements. Often, the soundness can also be regarded as a stricter notion of unforgeability for file tags to avoid cheating the ownership. This means that the CSPs, even if collusion is attempted, cannot be tampered with the data or forge the data tags if the soundness property holds. Thus, the Theorem 3 denotes that the CPDP scheme can resist the *tag forgery attacks* (see Attacks 2 and 4 in Appendix A) to avoid cheating the CSPs' ownership.

5 Performance Evaluation

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

1. It is a proof method in which a proposition is proved to be true by proving that it is impossible to be false.

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 $[B]$).

TABLE 3
Comparison of computation overheads between our CPDP scheme and non-cooperative (trivial) scheme.

	CPDP Scheme	Trivial Scheme
KeyGen	$3[E]$	2[E]
TagGen	$(2n+s)[E]$	$(2n+s)[E]$
Proof($\mathcal{P}$)	$c[B]+(t+cs+1)[E]$	$c[B]+(t+cs-c)[E]$
Proof(V)	$3[B]+(t+s)[E]$	$3c[B]+(t+cs)[E]$

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 non-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 multi-cloud. 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.

TABLE 4
Comparison of communication overheads between our CPDP and non-cooperative (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	$sl_0+l_1+l_T$	

Without loss of generality, let the security parameter κ be 80 bits, we need the elliptic curve domain parameters over $\mathbb{F}_p$ with $|p| = 160$ bits 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 $\mathbb{G}_\mathbb{T}$ 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 $size(f) = 1M$-bytes is $store(f) = n \cdot s \cdot l_0 + n \cdot l_1 = 1.04M$-bytes for $n = 10^3$ and $s = 50$. The storage overhead of its index table χ is $n \cdot l_0 = 20K$-bytes. We define the overhead rate as $\lambda = \frac{store(f)}{size(f)} - 1 = \frac{l_1}{s \cdot l_0}$ 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 $s \cdot l_0 + l_1 + l_T \approx 1.3K$-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 $\rho_b = \frac{e}{n}$. Let t be the number of queried blocks for a challenge in the verification protocol. We have detection probability [2]

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

where, $P(\rho_b, t)$ denotes that the probability P is a function over ρ_b and t. Hence, the number of queried blocks is $t \approx \frac{\log(1-P)}{\log(1-\rho_b)} \approx \frac{P \cdot n}{e}$ for a sufficiently large n and $t \ll n$.[3] This means that the number of queried blocks t is directly proportional to the total number of file blocks n for the constant P and e. Therefore, for a uniform random verification in a PDP scheme with *fragment structure*, given a file with $sz = n \cdot s$ sectors and the probability of sector corruption ρ, the detection probability of verification protocol has $P \geq 1 - (1-\rho)^{sz \cdot \omega}$, where ω denotes the sampling probability in the verification protocol. We can obtain this result as follows: because $\rho_b \geq 1 - (1-\rho)^s$ is the probability of block corruption with s sectors in common PDP scheme, the verifier can detect block errors with probability $P \geq 1 - (1-\rho_b)^t \geq 1 - ((1-\rho)^s)^{n \cdot \omega} = 1 - (1-\rho)^{sz \cdot \omega}$ for a challenge with $t = n \cdot \omega$ index-coefficient pairs. In the same way, given a multi-cloud $\mathcal{P} = \{P_i\}_{i \in [1,c]}$, the detection probability of CPDP scheme has

$$\begin{aligned} & P(sz, \{\rho_k, r_k\}_{P_k \in \mathcal{P}}, \omega) \\ \geq\ & 1 - \prod\nolimits_{P_k \in \mathcal{P}} ((1-\rho_k)^s)^{n \cdot r_k \cdot \omega} \\ =\ & 1 - \prod\nolimits_{P_k \in \mathcal{P}} (1-\rho_k)^{sz \cdot r_k \cdot \omega}, \end{aligned}$$

where r_k denotes the proportion of data blocks in the k-th CSP, ρ_k denotes the probability of file corruption

2. Exactly, we have $P = 1 - (1-\frac{e}{n}) \cdot (1-\frac{e}{n-1}) \cdots (1-\frac{e}{n-t+1})$. Since $1 - \frac{e}{n} \geq 1 - \frac{e}{n-i}$ for $i \in [0, t-1]$, we have $P = 1 - \prod_{i=0}^{t-1}(1 - \frac{e}{n-i}) \geq 1 - \prod_{i=0}^{t-1}(1-\frac{e}{n}) = 1 - (1-\frac{e}{n})^t$.

3. In terms of $(1-\frac{e}{n})^t \approx 1 - \frac{e \cdot t}{n}$, we have $P \approx 1 - (1 - \frac{e \cdot t}{n}) = \frac{e \cdot t}{n}$.

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	87/236
0.95	3/8	11/29	31/86	100/267
0.99	4/10	13/31	39/105	119/345
0.999	5/11	16/38	48/128	146/433

in the k-th CSP, and $r_k \cdot \omega$ denotes the possible number of blocks queried by the verifier in the k-th CSP. Furthermore, we observe the ratio of queried blocks in the total file blocks w under different detection probabilities. Based on above analysis, it is easy to find that this ratio holds the equation

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

When this probability ρ_k is a constant probability, the verifier can detect sever misbehavior with a certain probability P by asking proof for the number of blocks $t \approx \frac{\log(1-P)}{s\log(1-\rho)}$ for PDP or

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

for CPDP, where $t = n \cdot w = \frac{sz \cdot w}{s}$. Note that, the value of t is dependent on the total number of file blocks n [2], because it is increased along with the decrease of ρ_k and $\log(1-\rho_k) < 0$ for the constant number of disrupted blocks e and the larger number n.

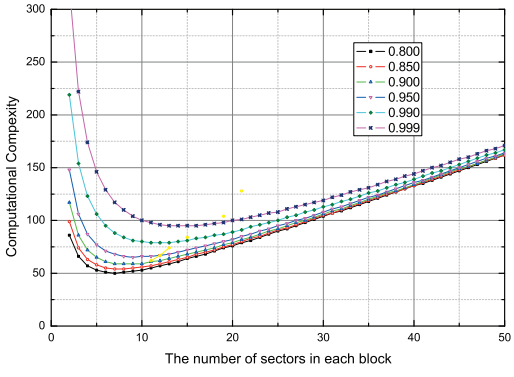


Fig. 4. The relationship between computational cost and the number of sectors in each block.

Another advantage of probabilistic verification based on random sampling is that it is easy to identify the tampering or forging data blocks or tags. The identification function is obvious: when the verification fails, we can choose the partial set 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 $O(\log n)$.

5.3 Parameter Optimization

In the fragment structure, the number of sectors per block s 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 can not 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 $a \cdot t + b \cdot s + c$ denotes the computational cost of verification protocol in PDP scheme, $a, b, c \in \mathbb{R}$, and c is a constant. This conclusion can be obtained from following process: Let $sz = n \cdot s = size(f)/l_0$. According to above-mentioned 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 k-th CSP, ρ_k denotes the probability of file corruption in the k-th CSP. Since $\frac{a}{s}$ is a monotone decreasing function and $b \cdot s$ is a monotone increasing function for $s > 0$, there exists an optimal value of $s \in \mathbb{N}$ 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 multi-cloud storage involves three CSPs $\mathcal{P} = \{P_1, P_2, P_3\}$ and the probability of sector corruption is a constant value $\{\rho_1, \rho_2, \rho_3\} = \{0.01, 0.02, 0.001\}$. 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

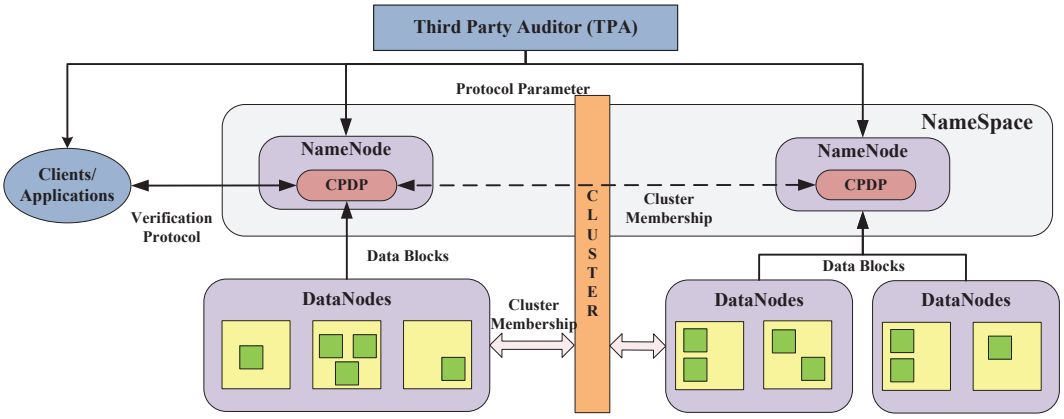


Fig. 5. Applying CPDP scheme in Hadoop distributed file system (HDFS).

proportion of data blocks is 50%, 30%, and 20% in three CSPs, respectively, that is, $r_1 = 0.5$, $r_2 = 0.3$, and $r_3 = 0.2$. 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 Figure 4. When s is less than the optimal value, the computational cost decreases evidently with the increase of s, and then it raises when s is more than the optimal value.

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

More accurately, we show the influence of parameters, $sz \cdot w$, s, and t, under different detection probabilities in Table 6. It is easy to see that computational cost raises with the increase of P. Moreover, we can make sure the sampling number of challenge with following conclusion: Given the detection probability P, the probability of sector corruption ρ, and the number of sectors in each block s, the sampling number of verification protocol are a constant $t = n \cdot w \geq \frac{\log(1-P)}{s \cdot \sum_{P_k \in \mathcal{P}} r_k \cdot \log(1-\rho_k)}$ for different files.

Finally, we observe the change of s under different ρ and P. The experimental results are shown in Table 5. It is obvious that the optimal value of s raises with increase of P and with the decrease of ρ. We choose the optimal value of s on the basis of practical settings and system requisition. For NTFS format, we suggest that the value of s is 200 and the size of block is 4K-Bytes, which is the same as the default size of cluster when the file size is less than 16TB in NTFS. In this case, the value of s ensures that the extra storage doesn't exceed 1% in storage servers.

5.4 CPDP for Integrity Audit Services

Based on our CPDP scheme, we introduce an audit system architecture for outsourced data in multiple clouds by replacing the TTP with a third party auditor (TPA) in Figure 1. In this architecture, this architecture can be constructed into a visualization infrastructure of cloud-based storage service [1]. In Figure 5, we show an example of applying our CPDP scheme in Hadoop distributed file system (HDFS) [4], which a distributed, scalable, and portable file system [19]. HDFS' architecture is composed of NameNode and DataNode, where NameNode maps a file name to a set of indexes of blocks and DataNode indeed stores data blocks. To support our CPDP scheme, the index-hash hierarchy and the metadata of NameNode should be integrated together to provide an enquiry service for the hash value $\xi_{i,k}^{(3)}$ or index-hash record χ_i. Based on the hash value, the clients can implement the verification protocol via CPDP services. Hence, it is easy to replace the checksum methods with the CPDP scheme for anomaly detection in current HDFS.

To validate the effectiveness and efficiency of our proposed approach for audit services, we have implemented a prototype of an audit system. We simulated the audit service and the storage service by using two local IBM servers with two Intel Core 2 processors at 2.16 GHz and 500M RAM running Windows Server 2003. These servers were connected via 250 MB/sec of network bandwidth. Using GMP and PBC libraries, we have implemented a cryptographic library upon which our scheme can be constructed. This C library contains approximately 5,200 lines of codes and has been tested on both 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 $|p| = 160$.

4. Hadoop can enable applications to work with thousands of nodes and petabytes of data, and it has been adopted by currently mainstream cloud platforms from Apache, Google, Yahoo, Amazon, IBM and Sun.

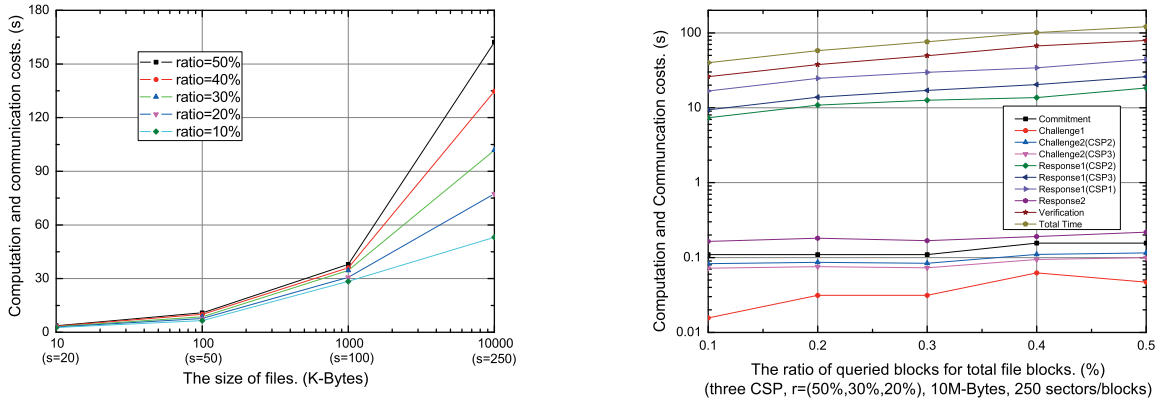


Fig. 6. Experimental results under different file size, sampling ratio, and sector number.

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 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 10KB to 10MB; the sector numbers were changed from 20 to 250 in terms of file sizes; and the sampling ratios were changed from 10% to 50%. The experimental results are shown in the left side of Figure 6. These results dictate that the computation and communication costs (including I/O costs) grow with the increase of file size and sampling ratio.

Next, we compare the performance of each activity in our verification protocol. We have shown the theoretical results in Table 4: the overheads of "commitment" and "challenge" resemble one another, and the overheads of "response" and "verification" resemble one another as well. To validate the theoretical results, we changed the sampling ratio w from 10% to 50% for a 10MB file and 250 sectors per block in a multi-cloud $\mathcal{P} = \{P_1, P_2, P_3\}$, in which the proportions of data blocks are 50%, 30%, and 20% in three CSPs, respectively. In the right side of Figure 6, our experimental results show that the computation and communication costs of "commitment" and "challenge" are slightly changed along with the sampling ratio, but those for "response" and "verification" grow with the increase of the sampling ratio. Here, "challenge" and "response" can be divided into two sub-processes: "challenge1" and "challenge2", as well as "response1" and "response2", respectively. Furthermore, the proportions of data blocks in each CSP have greater influence on the computation costs of "challenge" and "response" processes. In summary, our scheme has better performance than non-cooperative approach.

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 CPDP scheme, 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 a issue to provide the support of variable-length block verification.

Acknowledgments

The work of Y. Zhu and M. Yu was supported by the National Natural Science Foundation of China (Project No.61170264 and No.10990011). This work of Gail-J.

Ahn and Hongxin Hu was partially supported by the grants from US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308).

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, 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," in *ACM Conference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 598–609.

[3] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 584–597.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks, SecureComm*, 2008, pp. 1–10.

[5] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 213–222.

[6] H. Shacham and B. Waters, "Compact proofs of retrievability," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350. Springer, 2008, pp. 90–107.

[7] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *ESORICS*, ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789. Springer, 2009, pp. 355–370.

[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," in *SAC*, W. C. Chu, W. E. Wong, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2011, pp. 1550–1557.

[9] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 187–198.

[10] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *TCC*, ser. Lecture Notes in Computer Science, O. Reingold, Ed., vol. 5444. Springer, 2009, pp. 109–127.

[11] L. Fortnow, J. Rompel, and M. Sipser, "On the power of multiprover interactive protocols," in *Theoretical Computer Science*, 1988, pp. 156–161.

[12] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han, and S. Chen, "Collaborative integrity verification in hybrid clouds," in *IEEE Conference on the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, Orlando, Florida, USA, October 15-18*, 2011, pp. 197–206.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep., Feb 2009.

[14] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (CRYPTO'2001)*, vol. 2139 of LNCS, 2001, pp. 213–229.

[15] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[16] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Des. Codes Cryptography*, vol. 42, no. 3, pp. 239–271, 2007.

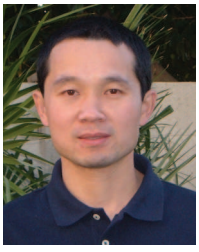
[17] J.-L. Beuchat, N. Brisebarre, J. Detrey, and E. Okamoto, "Arithmetic operators for pairing-based cryptography," in *CHES*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, 2007, pp. 239–255.

[18] H. Hu, L. Hu, and D. Feng, "On a class of pseudorandom sequences from elliptic curves over finite fields," *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2598–2605, 2007.

[19] A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley, "Hadoop: A framework for running applications on large clusters built of commodity hardware," Tech. Rep., 2005. [Online]. Available: http://lucene.apache.org/hadoop/

[20] E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds., *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, 2009.

Yan Zhu received the Ph.D. degree in computer science from Harbin Engineering University, China, in 2005. He was an associate professor of computer science in the Institute of Computer Science and Technology at Peking University since 2007. He worked at the Department of Computer Science and Engineering, Arizona State University as a visiting associate professor from 2008 to 2009. His research interests include cryptography and network security.



Hongxin Hu is currently working toward the Ph.D. degree from the School of Computing, Informatics, and Decision Systems Engineering, Ira A. Fulton School of Engineering, Arizona State University. He is also a member of the Security Engineering for Future Computing Laboratory, Arizona State University. His current research interests include access control models and mechanisms, security and privacy in social networks, and security in distributed and cloud computing, network and system security and secure software engineering.

Gail-Joon Ahn is an Associate Professor in the School of Computing, Informatics, and Decision Systems Engineering, Ira A. Fulton Schools of Engineering and the Director of Security Engineering for Future Computing Laboratory, Arizona State University. His research interests include information and systems security, vulnerability and risk management, access control, and security architecture for distributed systems, which has been supported by the U.S. National Science Foundation, National Security Agency, U.S. Department of Defense, U.S. Department of Energy, Bank of America, Hewlett Packard, Microsoft, and Robert Wood Johnson Foundation. Dr. Ahn is a recipient of the U.S. Department of Energy CAREER Award and the Educator of the Year Award from the Federal Information Systems Security Educators Association. He was an Associate Professor at the College of Computing and Informatics, and the Founding Director of the Center for Digital Identity and Cyber Defense Research and Laboratory of Information Integration, Security, and Privacy, University of North Carolina, Charlotte. He received the Ph.D. degree in information technology from George Mason University, Fairfax, VA, in 2000.

Mengyang Yu received his B.S. degree from the School of Mathematics Science, Peking University in 2010. He is currently a M.S. candidate in Peking University. His research interests include cryptography and computer security.