The Journal of Systems and Software 85 (2012) 1083–1095

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Efficient audit service outsourcing for data integrity in clouds

Yan Zhu[a,b,*], Hongxin Hu[c], Gail-Joon Ahn[c], Stephen S. Yau[c]

[a] Institute of Computer Science and Technology, Peking University, Beijing 100871, China
[b] Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing 100871, China
[c] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, ZA 85281, USA

ARTICLE INFO

Article history:
Received 31 March 2011
Received in revised form
21 September 2011
Accepted 10 December 2011
Available online 19 December 2011

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

ABSTRACT

Cloud-based outsourced storage relieves the client's burden for storage management and maintenance by providing a comparably low-cost, scalable, location-independent platform. However, the fact that clients no longer have physical possession of data indicates that they are facing a potentially formidable risk for missing or corrupted data. To avoid the security risks, audit services are critical to ensure the integrity and availability of outsourced data and to achieve digital forensics and credibility on cloud computing. Provable data possession (PDP), which is a cryptographic technique for verifying the integrity of data without retrieving it at an untrusted server, can be used to realize audit services.

In this paper, profiting from the interactive zero-knowledge proof system, we address the construction of an interactive PDP protocol to prevent the fraudulence of prover (soundness property) and the leakage of verified data (zero-knowledge property). We prove that our construction holds these properties based on the computation Diffie–Hellman assumption and the rewindable black-box knowledge extractor. We also propose an efficient mechanism with respect to probabilistic queries and periodic verification to reduce the audit costs per verification and implement abnormal detection timely. In addition, we present an efficient method for selecting an optimal parameter value to minimize computational overheads of cloud audit services. Our experimental results demonstrate the effectiveness of our approach.

1. Introduction

In recent years, the emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional information technology. Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. One fundamental aspect of this paradigm shifting is that data are being centralized and outsourced into clouds. This kind of outsourced storage services in clouds have become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data.

The cloud storage service (CSS) relieves the burden of storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to users since their data or archives are stored into an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices. However, they are still susceptible to security threats both from outside and inside the cloud (Armbrust et al., 2010); for the benefits of their possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users (Tchifilionova, 2011); furthermore, the dispute occasionally suffers from the lack of trust on CSP. Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users' own improper operations (Ko et al., 2011). Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of the stored data (Yavuz and Ning, 2009).

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

* Corresponding author at: Institute of Computer Science and Technology, Peking University, Beijing 100871, China. Tel.: +86 13121328791; fax: +86 10 82529207.
E-mail addresses: yan.zhu@pku.edu.cn (Y. Zhu), hxhu@asu.edu (H. Hu), gahn@asu.edu (G.-J. Ahn), yau@asu.edu (S.S. Yau).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.12.024

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

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

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

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

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

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

1.1. Contributions

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

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

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

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

1.2. Organization

This paper is organized as follows: in Section 2, we describe an audit architecture and security requirements of cloud audit systems. Section 3 introduces our audit scheme and analyzes the security of our scheme. In Section 4, we analyze the audit performance based on probabilistic queries and periodic verification. Section 5 gives an optimization algorithm of tag storage and verification protocol. Our implementation and experimental results are described in Section 6. Section 7 overviews the related work and we conclude this paper in Section 8.

2. Audit system architecture

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

Data owner (DO): who has a large amount of data to be stored in the cloud;

Cloud service provider (CSP): who provides data storage service and has enough storage spaces and computation resources;

Third party auditor (TPA): who has capabilities to manage or monitor outsourced data under the delegation of data owner; and

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

Scheme	CSP computation	Client computation	Communication	Fragment structure	Privacy	Ownership proof	Prob. of detection
PDP (Ateniese et al., 2007)	$O(t)$	$O(t)$	$O(1)$		√	√	$1-(1-\rho_b)^t$
SPDP (Ateniese et al., 2008)	$O(t)$	$O(t)$	$O(t)$		√		$1-(1-\rho_b)^t$
DPDP-I (Erway et al., 2009)	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$				$1-(1-\rho_b)^t$
DPDP-II (Erway et al., 2009)	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$				$1-(1-\rho_b)^{\Omega(n)}$
CPOR-I (Shacham and Waters, 2008)	$O(t)$	$O(t)$	$O(1)$		√	√	$1-(1-\rho_b)^t$
CPOR-II (Shacham and Waters, 2008)	$O(t+s)$	$O(t+s)$	$O(s)$	√		√	$1-(1-\rho)^{t\cdot s}$
Our scheme	$O(t+s)$	$O(t+s)$	$O(s)$	√	√	√	$1-(1-\rho)^{t\cdot s}$

Note that s is the number of sectors in each block, t is the number of sampling blocks, and ρ_b, ρ are the probability of block and sector corruption in a cloud server, respectively.

Granted applications (GA): who have the right to access and manipulate stored data. These applications can be either inside clouds or outside clouds according to the specific requirements.

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

- First, the client (data owner) uses the secret key sk to pre-processes the file, which consists of a collection of n blocks, generates a set of public verification information that is stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy;
- At a later time, using a protocol of proof of retrievability, TPA (as an audit agent of clients) issues a challenge to audit (or check) the integrity and availability of the outsourced data in terms of the public verification information. It is necessary to give an alarm for abnormal events.

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

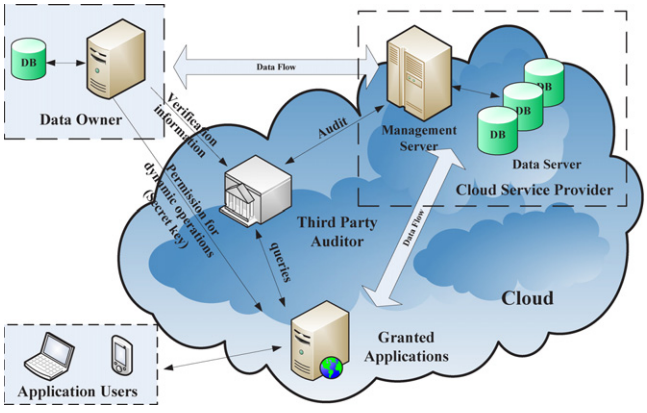

Fig. 1. Audit system architecture for cloud computing.

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

In this audit architecture, our core idea is to maintain the security of TPA to guarantee the credibility of cloud storages. This is because it is more easy and feasible to ensure the security of one TTP than to maintain the credibility of the whole cloud. Hence, the TPA could be considered as the root of trust in clouds.

To enable privacy-preserving public auditing for cloud data storage under this architecture, our protocol design should achieve following security and performance guarantees:

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

Verification-correctness: to ensure there exists no cheating CSP that can pass the audit from TPA without indeed storing users' data intact;

Privacy-preserving: to ensure that there exists no way for TPA to derive users' data from the information collected during the auditing process; and

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

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

3. Construction of interactive audit scheme

In this section, we propose a cryptographic interactive audit scheme (also called as interactive PDP, IPDP) to support our audit system in clouds. This scheme is constructed on the standard model of interactive proof system, which can ensure the confidentiality of

secret data (zero-knowledge property) and the undeceivability of invalid tags (soundness property).

3.1. Notations and preliminaries

Let $\mathcal{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ indexed by $k \in \mathcal{K}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in breaking the collision-resistance of $\mathcal{H}$ 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 random choice of $k \in \mathcal{K}$ and random bits of $\mathcal{A}$. This hash function can be obtained from hash function of BLS signatures (Boneh et al., 2004).

Definition 1 *(Collision-resistant hash).* A hash family $\mathcal{H}$ is (t, ϵ)-collision-resistant if no t-time adversary has advantage at least ϵ in breaking the collision-resistance of $\mathcal{H}$.

We set up our systems using bilinear pairings proposed by Boneh and Franklin (2001). Let $\mathbb{G}$ be two multiplicative groups using elliptic curve conventions with large prime order p. The function e be a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with 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 $H = 1$. (3) Computability: $e(G, H)$ is efficiently computable.

Definition 2 *(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. Definition of interactive audit

We present a definition of interactive audit protocol based on interactive proof systems as follows:

Definition 3. A cryptographic interactive audit scheme $\mathcal{S}$ is a collection of two algorithms and an interactive proof system, $\mathcal{S} = (\mathcal{K}, \mathcal{T}, \mathcal{P})$:

- $\mathcal{K}eyGen(1^s)$: takes a security parameter s as input, and returns a public-secret keypair (pk, sk);
- $\mathcal{T}agGen(sk, F)$: takes as inputs the secret key sk and a file F, and returns the triples (ζ, ψ, σ), where ζ denotes the secret used to generate verification tags, ψ is the set of public verification parameters u and index information χ, i.e., $\psi = (u, \chi)$; σ denotes the set of verification tags;
- $\mathcal{P}roof(CSP, TPA)$: is a public two-party proof protocol of retrievability between CSP (prover) and TPA (verifier), that is $\langle CSP(F, \sigma), TPA \rangle (pk, \psi)$, where CSP takes as input a file F and a set of tags σ, and a public key pk and a set of public parameters ψ are the common input between CSP and TPA. At the end of the protocol run, TPA returns $\{0|1\}$, where 1 means the file is correct stored on the server.

where, $P(x)$ denotes the subject P holds the secret x and $\langle P, V \rangle (x)$ denotes both parties P and V share a common data x in a protocol.

This is a more generalized model than existing verification models for outsourced data. Since the verification process is considered as an interactive protocol, this definition does not limit to the specific steps of verification, including scale, sequence, and the number of moves in protocol, so it can provide greater convenience for the construction of protocol.

KeyGen(1^κ)**:** Let $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear map group system with randomly selected generators $g, h \in_R \mathbb{G}$, where $\mathbb{G}$, $\mathbb{G}_T$ are two group of large prime order p, $|p| = O(\kappa)$. Generate a collision-resistant hash function $H_k(\cdot)$ and chooses a random $\alpha, \beta \in_R \mathbb{Z}_p$ and computes $H_1 = h^\alpha$ and $H_2 = h^\beta \in \mathbb{G}$. Thus, the secret key is $sk = (\alpha, \beta)$ and the public key is $pk = (g, h, H_1, H_2)$.

TagGen(sk, F)**:** Splits the file F into $n \times s$ sectors $F = \{m_{i,j}\} \in \mathbb{Z}_p^{n \times s}$. 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 $i \in [1, s]$ and $\xi^{(1)} = H_\xi(\text{“}Fn\text{”})$, where $\xi = \sum_{i=1}^{s} \tau_i$ and Fn is the file name. Builds an index table $\chi = \{\chi_i\}_{i=1}^n$, then calculates its tag as

$$\sigma_i \leftarrow (\xi_i^{(2)})^\alpha \cdot g^{\sum_{j=1}^{s} \tau_j \cdot m_{i,j} \cdot \beta} \in \mathbb{G}.$$

where $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ and $i \in [1, n]$. Finally, sets $u = (\xi^{(1)}, u_1, \cdots, u_s)$ and outputs $\zeta = (\tau_1, \cdots, \tau_s)$, $\psi = (u, \chi)$ to TTP, and $\sigma = (\sigma_1, \cdots, \sigma_n)$ to CSP.

Proof(CSP, TPA)**:** This is a 3-move protocol between CSP and TPA with the common input (pk, ψ), as follows:

- **Commitment**$(CSP \to TPA)$: CSP chooses a random $\gamma \in \mathbb{Z}_p$ and s random $\lambda_j \in_R \mathbb{Z}_p$ for $j \in [1, s]$, and sends its commitment $C = (H_1', \pi)$ to TPA, where $H_1' = H_1^\gamma$ and $\pi \leftarrow e(\prod_{j=1}^{s} u_j^{\lambda_j}, H_2)$;
- **Challenge**$(CSP \leftarrow TPA)$: TPA chooses a random challenge set I of t indexes along with t random coefficients $v_i \in \mathbb{Z}_p$. Let Q be the set $\{(i, v_i)\}_{i \in I}$ of challenge index coefficient pairs. TPA sends Q to CSP;
- **Response**$(CSP \to TPA)$: CSP calculates the response θ, μ as

$$\begin{cases} \sigma' \leftarrow \prod_{(i,v_i) \in Q} \sigma_i^{\gamma \cdot v_i}, \\ \mu_j \leftarrow \lambda_j + \gamma \cdot \sum_{(i,v_i) \in Q} v_i \cdot m_{i,j}, \end{cases}$$

where $\mu = \{\mu_j\}_{j \in [1,s]}$. P sends $\theta = (\sigma', \mu)$ to V;

Verification: TPA can check that the response was correctly formed by checking that

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

Fig. 2. Proposed interactive audit protocol.

3.3. Proposed construction

We present our construction of audit scheme in Fig. 2. This scheme involves three algorithms: key generation, tag generation, and verification protocol. In the key generation algorithm, each client is assigned a secret key sk, which can be used to generate the tags of many files, and a public key pk, which be used to verify the integrity of stored files.

In tag generation algorithm, each processed file F will produce a public verification parameter $\psi = (u, \chi)$, where $u = (\xi^{(1)}, u_1, \ldots, u_s)$, $\chi = \{\chi_i\}_{i\in[1,n]}$ is a hash index table. The hash value $\xi^{(1)} = H_\xi(\text{“}Fn\text{”})$ can be considered as the signature of the secret $\tau_1, \ldots, \tau_s$ and $u_1, \ldots, u_s$ denotes the “encryption” of these secrets. The structure of hash index table should be designed according to applications. For example, for a static, archival file, we can define briefly $\chi_i = B_i$, where B_i is the sequence number of block; for a dynamic file, we can also define $\chi_i = (B_i||V_i||R_i)$, where B_i is the sequence number of block, R_i is the version number of updates for this block, and R_i is a random integer to avoid collision. The index table χ is very important to ensure the security of files. According to χ and $\xi^{(1)}$, we can generate the hash value $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ for each block. Note that, it must assure that the ψ's are different for all processed files.

In our construction, the verification protocol has a 3-move structure: commitment, challenge and response, which is showed in Fig. 3. This protocol is similar to Schnorr's Σ protocol (Schnorr, 1991), which is a zero-knowledge proof system. Using this property, we ensure the verification process does not reveal anything other than the veracity of the statement of data integrity in a private cloud. In order to prevent the leakage of data and tags in the verification process, the secret data $\{m_{i,j}\}$ are protected by a random $\lambda_j \in \mathbb{Z}_p$ and the tags $\{\sigma_i\}$ are randomized by a $\gamma \in \mathbb{Z}_p$. Furthermore, the values $\{\lambda_j\}$ and γ are protected by the simple commitment methods, i.e., H_1^γ and $u_i^{\lambda_i} \in \mathbb{G}$, to avoid the adversary from gaining them.

3.4. Security analysis

According to the standard model of interactive proof system proposed by Bellare and Goldreich (Goldreich, 2001), the proposed protocol $\mathcal{P}roof(CSP, TPA)$ has completeness, soundness, and zero-knowledge properties described below.

3.4.1. Completeness property

For every available tag $\sigma \in TagGen(sk, F)$ and a random challenge $Q = (i, v_i)_{i\in I}$, the completeness of protocol can be elaborated as follows:

$$\pi \cdot e(\sigma', h) = e(g, h)^{\beta\sum_{j=1}^{s}\tau_j\cdot\lambda_j} \cdot e\left(\prod_{(i,v_i)\in Q}(\xi_i^{(2)})^{v_i}, h\right)^{\alpha\cdot\gamma}$$

$$\cdot e(g, h)^{\gamma\cdot\beta\sum_{j=1}^{s}(\tau_j\cdot\sum_{(i,v_i)\in Q}v_i\cdot m_{i,j})}$$

$$= e\left(\prod_{(i,v_i)\in Q}(\xi_i^{(2)})^{v_i}, h^{\alpha\cdot\gamma}\right)\cdot\prod_{j=1}^{s}e(u_j^{\mu_j}, h^\beta).$$

This equation means that the proposed protocol is efficient for valid tags.

3.4.2. Soundness property

The soundness property means that it be infeasible to fool the verifier into accepting false statements. We can define the soundness of our protocol as follows: for every faked $\sigma^* \notin TagGen(sk, F)$ and every (really or deceptively) interactive machine P^*,

$$Pr[\langle CSP^*(F, \sigma^*), TPA\rangle(pk, \psi) = 1] \le 1/p(\kappa); \quad (1)$$

where, $p(\cdot)$ is one polynomial and κ is a security parameter used in $KeyGen(1^\kappa)$. For every tag $\sigma^* \notin TagGen(sk, F)$, we assume that there exists an interactive machine P^* can pass verification with any verifier V^* with noticeable probability.

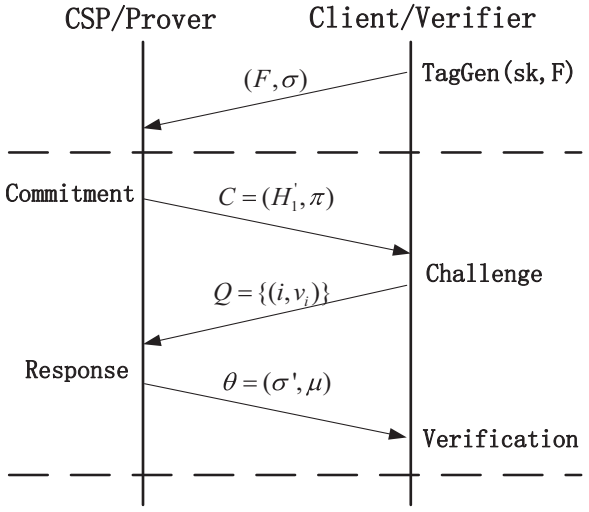

Fig. 3. Framework of interactive audit scheme.

In order to prove the nonexistence of P^*, to the contrary, we make use of P^* to construct a knowledge extractor $\mathcal{M}$, which gets the common input (pk, ψ) and rewindable black-box access to the prover P^* and attempts to break the computation Diffie–Hellman (CDH) assumption in $\mathbb{G}$: given G, $G_1 = G^a$, $G_2 = G^b \in_R \mathbb{G}$, output $G^{ab} \in \mathbb{G}$. We have following theorem (the proof is described in Appendix A):

Theorem 1. *Our audit protocol has (t, ϵ') knowledge soundness in random oracle and rewindable knowledge extractor model assuming the (t, ϵ)-computation Diffie–Hellman (CDH) assumption holds in $\mathbb{G}$ for $\epsilon' \ge \epsilon$.*

The soundness can also be regarded as a stricter notion of unforgeability for file tags. Thus, this theorem means that the prover cannot forge file tags or tamper with the data.

3.4.3. Zero-knowledge property

In order to protect the confidentiality of checked data, we are more concerned about the leakage of private information in the verification process. It is easy to find that data blocks and their tags could be obtained by the verifier in some existing schemes. To solve this problem, we introduce zero-knowledge property into our audit system as follows: an interactive proof of retrievability scheme is computational zero knowledge if there exists a probabilistic polynomial-time algorithm S^* (call a simulator) such that for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently large s, it holds that

$$\Big|Pr[D(pk, \psi, S^*(pk, \psi)) = 1]$$

$$-Pr[D(pk, \psi, \langle CSP(F, \sigma), TPA^*\rangle(pk, \psi)) = 1]\Big| \le \frac{1}{p(\kappa)},$$

where, $S^*(pk, \psi)$ denotes the output of simulator S on common input (pk, ψ) and $\langle CSP(F, \sigma), TPA^*\rangle(pk, \psi)$[1] denotes the view of interactive protocol between TPA^* and $CSP(F, \sigma)$ on common input (pk, ψ). This equation implies that, for all $\sigma \in TagGen(sk, F)$, the ensembles $S^*(pk, \psi)$ and $\langle CSP(F, \sigma), TPA^*\rangle(pk, \psi)$ are computationally indistinguishable.

[1] It can also be written as $View(\langle CSP(F, \sigma), TPA^*\rangle(pk, \psi))$.

Table 2
Signal and its explanation.

Signal	Description
ρ	The probability of sector corruption
ρ_b	The probability of block corruption
P	The detection probability at each verification
P_T	The total detection probability in an audit period
w	The ratio of disrupted blocks in total file blocks
f	The frequency of verification
n	The number of file blocks
s	The number of sectors in each block
sz	The total size of outsourced file
t	The sampling number in verification
e	The number of disrupted blocks

Actually, zero-knowledge is a property that captures *CSP*'s robustness against attempts to gain knowledge by interacting with it. For our audit scheme, we make use of the zero-knowledge property to guarantee the security of data blocks and signature tags.

Theorem 2. *The verification protocol $\mathcal{P}roof(CSP, TPA)$ has the computational zero-knowledge property in our interactive audit scheme.*

Using the standard simulator $S^*(pk, \psi)$, the proof of this theorem is described in Appendix B.

4. Optimizing the schedule for probabilistic verifications

No doubt too frequent audit activities will increase the computation and communication overheads, but less frequent activities may not detect abnormality timely. Hence, the scheduling of audit activities is important for improving the quality of audit services.

In order to detect abnormality in a low-overhead and timely manner, we optimize the performance of audit systems from two aspects: performance evaluation of probabilistic queries and schedule of periodic verification. Our basis thought is to achieve overhead balancing by verification dispatching, which is one of the efficient strategies for improving the performance of audit systems.

For clarity, we list some used signals in Table 2.

4.1. Performance evaluation of probabilistic queries

The audit service achieves the detection of CSP servers misbehavior in a random sampling mode in order to reduce the workload on the server. The detection probability P of disrupted blocks is an important parameter to guarantee that these blocks can be detected in time. Assume the TPA modifies e blocks out of the n-block file. The probability of disrupted blocks is $\rho_b = e/n$. Let t be the number of queried blocks for a challenge in the protocol proof. We have detection probability

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

Hence, the number of queried blocks is $t = \log(1-P)/\log(1-\rho_b)$.

In Fig. 4, we show the same result for the number of queried blocks under different detection probabilities (from 0.5 to 0.99), the different number of file blocks (from 200 to 10,000), and the constant number of disrupted blocks (100). It is easy to find that the number of queried blocks t is directly proportional to the total number of file blocks n for the constant P and e, that is, $t \approx c(P \cdot n)/e$ for a sufficiently large n, where c is a constant.[2]

[2] In terms of $(1-e/n)^t \approx 1-(e\cdot t)/n$, we have $P \approx 1-(1-(e\cdot t)/n) = (e\cdot t)/n$.

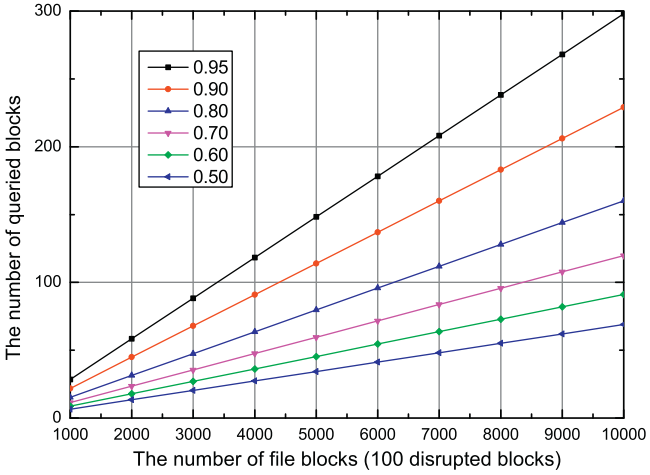

Fig. 4. Numbers of queried blocks under different detection probabilities and the different number of file blocks.

Furthermore, we observe the ratio of queried blocks in the total file blocks $w = t/n$ under different detection probabilities. Based on above analysis, it is easy to find that this ratio holds the equation

$$w = \frac{t}{n} = \frac{\log(1-P)}{n \cdot \log(1-\rho_b)}.$$

To clearly represent this ratio, Fig. 5 plots r for different values of n, e and P. It is obvious that the ratio of queried blocks tends to be a constant value for a sufficiently large n. For instance, in Fig. 5, if there exist 10 disrupted blocks, the TPA asks for $w = 30$ and 23% of n ($1000 \le n \le 10,000$) in order to achieve P of at least 95% and 90%, respectively. Moreover, this ratio w is also inversely proportional to the number of disrupted blocks e. For example, if there exist 100 disrupted blocks, the TPA needs merely to ask for $w = 4.5$ and 2.3% of n ($n > 1000$) in order to achieve the same P, respectively. Hence, the audit scheme is very effective for higher probability of disrupted blocks.

In most cases, we adopt the probability of disrupted blocks to describe the possibility of data loss, damage, forgery or unauthorized changes. When this probability ρ_b is a constant probability, the TPA can detect sever misbehaviors with a certain probability P by asking proof for a constant amount of blocks $t = \log(1-P)/\log(1-\rho_b)$, independently of the total number of file

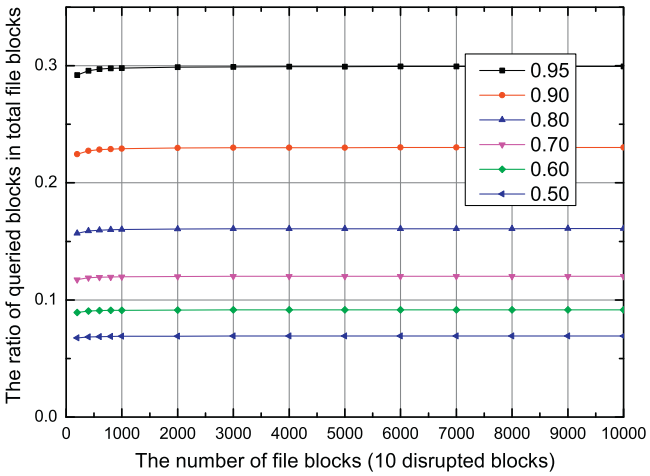

Fig. 5. Ratio of queried blocks in the total file blocks under different detection probabilities and different number of disrupted blocks (for 10 disrupted blocks).

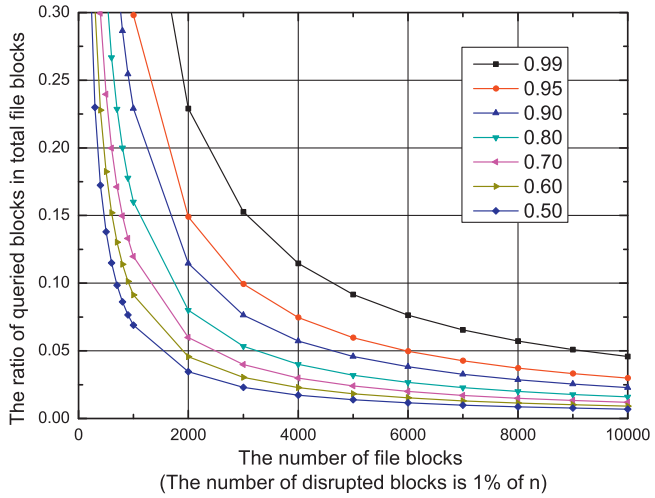

Fig. 6. Ratio of queried blocks in total file blocks under different detection probabilities and 1% disrupted blocks.

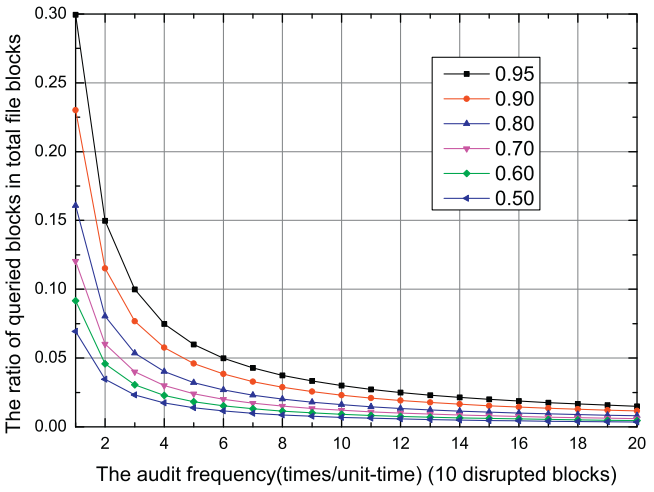

Fig. 7. Ratio of queried blocks in the total file blocks under different audit frequency for 10 disrupted blocks and 10,000 file blocks.

blocks (Ateniese et al., 2007). In Fig. 6, we show the ratio changes for different detection probabilities under 1% disrupted blocks, e.g., the TPA asks for 458, 298 and 229 blocks in order to achieve P of at least 99%, 95% and 90%, respectively. This kind of constant ratio is useful for the uniformly distributed ρ_b, especially for the storage device's physical failures.

4.2. Schedule of periodic verification

Clearly, too frequent audits would lead to a waste of network bandwidth and computing resources of TPA, Clients, and CSPs. On the other hand, too loose audits are not conducive to detect the exceptions in time. For example, if a data owner authorizes TPA to audit the data once a week, TPA arranges this task at a fixed time on each weekend. A malicious attack may be implemented after finishing an audit period, then there is enough time for the attacker to destroy all evidences and escape punishments. Thus, it is necessary to disperse the audit tasks throughout the entire audit cycle so as to balance the overload and increase the difficulty of malicious attacks.

Sampling-based audit has the potential to greatly reduce the workload on the servers and increase the audit efficiency. Firstly, we assume that each audited file has a audit period T, which depends on how important it is for the owner. For example, the common audit period may be assigned as 1 week or 1 month, and the audit period for important files may be set as 1 day. Of course, these audit activities should be carried out as night or on weekends.

Assume we make use of the audit frequency f to denote the number of occurrences of an audit event per unit time. This means that the number of TPA's queries is $T \cdot f$ times in an audit period T. According to above analysis, we have the detection probability $P = 1 - (1 - \rho_b)^{n \cdot w}$ in each audit event. Let P_T denote the detection probability in an audit period T. Therefore, we have the equation $P_T = 1 - (1 - P)^{T \cdot f}$. In terms of $1 - P = (1 - \rho_b)^{n \cdot w}$, the detection probability P_T can be denoted as

$$P_T = 1 - (1 - \rho_b)^{n \cdot w \cdot T \cdot f}.$$

In this equation, TPA can obtain the probability ρ_b depending on the transcendental knowledge for the cloud storage provider. Moreover, the audit period T can be appointed by a data owner in advance. Hence, the above equation can be used to analyze the parameter value w and f. It is obvious to obtain the equation

$$f = \frac{\log(1 - P_T)}{w \cdot n \cdot T \cdot \log(1 - \rho_b)}.$$

This means that the audit frequency f is inversely proportional to the ratio of queried blocks w. That is, with the increase of verification frequency, the number of queried blocks decreases at each verification process. In Fig. 7, we show the relationship between f and w under 10 disrupted blocks for 10,000 file blocks. It is easy to find that a marked drop of w after the increase of frequency.

In fact, this kind of relationship between f and w is a comparatively stable value for certain P_T, ρ_b, and n due to $f \cdot w = (\log(1 - P_T))/(n \cdot T \cdot \log(1 - \rho_b))$. TPA should choose appropriate frequency to balance the overhead according to above equation. For example, if $e = 10$ blocks in 10,000 blocks ($\rho_b = 0.1\%$), then TPA asks for 658 blocks or 460 blocks for $f = 7$ or 10 in order to achieve P_T of at least 99%. Hence, appropriate audit frequency would greatly reduced the sampling numbers, thus, the computation and communication overheads will also be reduced.

5. Optimization of tag storage and verification protocol

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. Thus, we propose an optimization algorithm for the value of s in this section. Our results show that the optimal value can not only minimize computation and communication overheads, but also reduce the size of extra storage, which is required to store verification tags in CSP.

5.1. Analysis of audit algorithm

We first analyze the computation cost of the audit scheme shown in Table 3. In this table, we use $[E]$ to denote the computation costs of an exponent operation in $\mathbb{G}$, namely, g^x, where x is a positive

Table 3
Performance analysis for our scheme.

	Our scheme
KeyGen	$2[E]$
TagGen	$(2n+s)[E]$
Proof(CSP)	$1[B]+(t+s+1)[E]$
Proof(TPA)	$3[B]+(t+s)[E]$

Table 4
Storage/communication overhead.

Algorithm		Our scheme
KeyGen	Client	$2l_0$
TagGen	CSP	$nsl_0 + nl_1$
	TPA	$sl_1 + (n+1)l_0$
Proof	Commit	$l_2 + l_T$
	Challenge	$2tl_0$
	Response	$sl_0 + (s+1)l_1$

integer in $\mathbb{Z}_p$ and $g \in \mathbb{G}$ or $\mathbb{G}_T$. We neglect the computation costs of algebraic operations and simple modular arithmetic operations because they run fast enough (Barreto et al., 2007). More complex operation is the computation of a bilinear map $e(\cdot,\cdot)$ between two elliptic points (denoted as $[B]$).

Secondly, we analyze the storage and communication costs of our scheme. We define the bilinear pairing taking the form $e : E(\mathbb{F}_{p^m}) \times E(\mathbb{F}_{p^{km}}) \to \mathbb{F}^*_{p^{km}}$ (the definition is from Beuchat et al., 2007; Hu et al., 2007), where p is a prime, m is a positive integer, and k is embedding degree (or security multiplier). In this case, we utilize asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ to replace symmetric pairing in the original schemes.

Without loss of generality, let the security parameter κ be 80-bits, we need 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 $\mathbb{G}_T$ for the embedding degree $k = 6$. Based on these definitions, we describe storage and communication costs in Table 4.

Hence, in our scheme, the storage overhead of a file with 1 MB is $n \cdot s \cdot l_0 + n \cdot l_1 = 1.04$ MB for $n = 10^3$ and $s = 50$, where $n \cdot s \cdot l_0 = 1$ MB is used to store the original file and $n \cdot l_1 = 40$ kB is the size of the stored tags. The storage overhead of its index table χ is $n \cdot l_0 = 20$ kB. Furthermore, in the verification protocol, the communication overhead of challenge is $2t \cdot l_0 = 40 \cdot t$-Bytes in terms of t, but its response has a constant-size communication overhead $s \cdot l_0 + (s+1) \cdot l_1 \approx 3$ kB for different-size files.

We define the overhead rate as $\lambda = store(f)/size(f) - 1 = \frac{l_1}{s \cdot l_0}$, where $store(f) = n \cdot s \cdot l_0 + n \cdot l_1$ and $size(f) = n \cdot s \cdot l_0$. And it should therefore be kept as low as possible in order to minimize the storage burden with respect to cloud storage providers. It is obvious that a larger s means more lower storage. However, the computational costs rise sharply when the number of s is large in terms of Table 3.

5.2. Optimization of parameters

We then turn our attention to the optimization of parameter s. Assume ρ denotes the probability of sector corruption. We have following theorem:

Given a file with $sz = n \cdot s$ sectors and the probability ρ of sector corruption, the detection probability of verification protocol has $P \geq 1 - (1-\rho)^{sz \cdot \omega}$, where $\omega = t/n$ denotes sampling probability in verification protocol. We can obtain following results: for a uniform random verification in the audit scheme, $\rho_b \geq 1 - (1-\rho)^s$ is the probability of block corruption with s sectors. For a challenge with $t = n \cdot \omega$ index-coefficient pairs, the verifier can detect block errors with probability $P \geq 1 - (1-\rho_b)^t \geq 1 - ((1-\rho)^s)^{n \cdot \omega} = 1 - (1-\rho)^{sz \cdot \omega}$, where $sz = n \cdot s$.

Given the detection probability P and the probability of sector corruption ρ, the optimal value of s is computed by

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

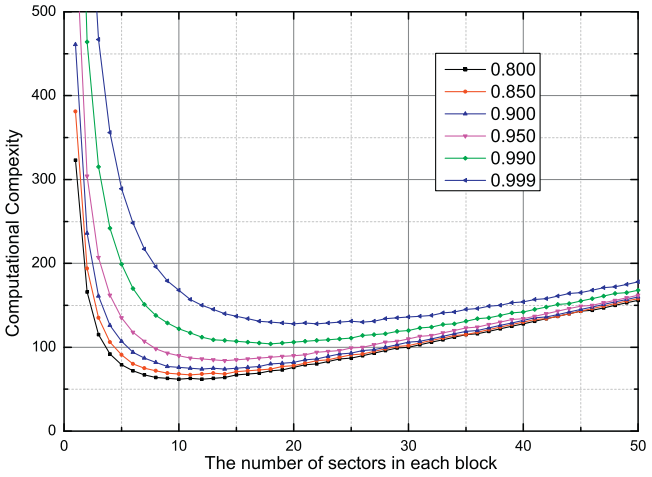

Fig. 8. Relationship between computational costs and the number of sectors in each block.

where $a \cdot t + b \cdot s + c$ denotes the computational cost of verification protocol in the audit scheme, $a, b, c \in \mathbb{R}$, and c is a constant. This conclusion can be obtained from the following process: let $sz = n \cdot s = size(f)/l_0$. According to above-mentioned results, the sampling probability holds $w \geq (\log(1-P))/(sz \cdot \log(1-\rho)) = (\log(1-P))/(n \cdot s \cdot \log(1-\rho))$. In order to minimize the computational cost, we have

$$\min_{s \in \mathbb{N}} \{a \cdot t + b \cdot s + c\} = \min_{s \in \mathbb{N}} \{a \cdot n \cdot w + b \cdot s + c\} \geq \min_{s \in \mathbb{N}} \left\{ \frac{\log(1-P)}{\log(1-\rho)} \frac{a}{s} + b \cdot s + c \right\}.$$

Since 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 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 the probability of sector corruption is a constant value $\rho = 0.01$. We set the detection probability P with a range from 0.8 to 1, e.g., $P = \{0.8, 0.85, 0.9, 0.95, 0.99, 0.999\}$. In terms of Table 3, the computational cost of the verifier can be simplified to $t + s$. Then, we can observe the computational costs under different s and P in Fig. 8. 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. More accurately, we show the influence of parameters, $sz \cdot w$, s, and t, under different detection probabilities in Table 5. It is easy to see that the computational costs raise with the increase of P. Moreover, we can make sure the sampling number of challenge with following conclusion: given P, ρ, and s, the sampling number of verification protocol are a constant $t = n \cdot w \geq (\log(1-P))/(s \cdot \log(1-\rho))$ for different files.

Finally, we observe the change of s under different ρ and P. Experimental results are showed in Table 6. It is obvious that the optimal value of s raises with the increase of P and with the decrease of ρ. We choose the optimal value of s on the basis of practical

Table 5
Influence of parameters under different detection probabilities P ($\rho = 0.01$).

P	0.8	0.85	0.9	0.95	0.99	0.999
$sz \cdot w$	160.14	188.76	229.11	298.07	458.21	687.31
s	12	14	14	14	18	25
t	14	14	17	22	26	28

Table 6
Influence of parameter s under different probabilities of corrupted blocks ρ and different detection probabilities P.

	0.1	0.01	0.001	0.0001
0.8	3	12	37	118
0.85	3	14	40	136
0.9	4	14	44	150
0.95	4	14	53	166
0.99	6	18	61	207
0.999	7	25	79	249

settings and system requisition. For NTFS format, we suggest that the value of s is 200 and the size of block is 4 kB, 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 does not exceed 1% in storage servers.

When the number of disrupted blocks is constant, we consider how the size of file sz influences on the optimal value of s. In Table 6, we can observe the change of values in each line grasps this influences. For example, the optimal value is changed from 79 to 249 if $\rho = 0.001$ and 0.0001 denote 1 disrupted blocks out of 1000 blocks and 10,000 blocks. The ratio between two values is 3.15. Furthermore, it is easy to find that the growth rate of s is 3.4 times when the size of file grows 10 times. Hence, we can also define that the value of s grows with the increase of the size of file and cluster in practical storage systems.

6. Implementation and experimental results

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

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

Firstly, we quantify the performance of our audit scheme under different parameters, such as file size sz, sampling ratio w, sector number per block s, and so on. Our previous analysis shows that the value of s should grow with the increase of sz in order to reduce computation and communication costs. Thus, our experiments were carried out as follows: the stored files were chosen from 10 KB to 10 MB, the sector numbers were changed from 20 to 250 in terms of the file sizes, and the sampling ratios were also changed from 10% to 50%. The experimental results were showed in the left side of Fig. 9. These results dictate that the computation and communication costs (including I/O costs) grow with increase of file size and sampling ratio.

Next, we compare the performance for each activity in our verification protocol. We have described the theoretical results in Tables 3 and 4: the overheads of "commitment" and "challenge" resemble one another, and the overheads of "response" and "verification" also resemble one another. To validate the theoretical results, we changed the sampling ratio w from 10% to 50% for a 10 MB file and 250 sectors per block. In the right side of Fig. 9, we show the experiment results, in which the computation and communication costs of "commitment" and "challenge" are slightly changed for sampling ratio, but those for "response" and "verification" grow with the increase of sampling ratio.

Finally, we evaluate the performance of our audit scheme in terms of computational overhead, due to these two schemes have constant-size communication overhead. For sake of comparison, our experiments used the same scenario as previous analysis, where a fix-size file is used to generate the tags and prove possession under different s. For a 150 kB file, the computational overheads of verification protocol are showed in Fig. 10(a) when the value of s ranges from 1 to 50 and the size of sector is 20-Bytes. It is obvious that the experiment result is consistent with the analysis in Fig. 8. The computational overheads of the tag generation are also showed in Fig. 10(b) in the same case. The results indicate that the overhead is reduced when the value of s is increased.

7. Related works

There has been a considerable amount of work done on untrusted outsourced storage. The most direct way to enforce the integrity control is to employ cryptographic hash function. Yumerefendi and Chase proposed a solution for authenticated network storage (Yumerefendi and Chase, 2007; Hsiao et al., 2009), using a hash tree (called as Merkle tree) as the underlying data structure. However their processing of updates is computationally expensive. Fu et al. (2002) described and implemented a method for efficiently and securely accessing a read-only file system that has been distributed to many providers. This architecture is a solution for efficiently authenticating operations on an outsourced file system.

Some recent work (Li et al., 2006; Ma et al., 2005; Xie et al., 2007; Yavuz and Ning, 2009) studied the problem of auditing the integrity for outsourced data or database. By explicitly assuming an order of the records in database, Pang et al. (Ma et al., 2005) used an aggregated signature to sign each record with the information from two neighboring records in the ordered sequence, which ensures the result of a simple selection query is continuous by checking the aggregated signature. Other work (Li et al., 2006; Xie et al., 2007) used a Merkle tree to audit the completeness of query results, but in some extreme cases, the overhead could be as high as processing these queries locally, which can significantly undermine the benefits of database outsourcing. Moreover, to ensure freshness, an extra system is needed to deliver the up-to-date root signature to all clients in a reliable and timely manner.

To check the integrity of stored data without download, some researchers have proposed two basic approaches called provable data possession (PDP) (Ateniese et al., 2007) and proofs of retrievability (POR) (Juels, 2007). Ateniese et al. (2007) first proposed the PDP model for ensuring possession of files on untrusted storages and provided an RSA-based scheme for the static case that achieves $O(1)$ communication costs. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the servers for data possession. This property greatly extend application areas of PDP protocol due to the separation of data owners and the authorized users.

In order to support dynamic data operations, Ateniese et al. have developed a dynamic PDP solution called scalable PDP (Ateniese et al., 2008). They proposed a lightweight PDP scheme based on cryptographic Hash function and symmetric key encryption, but

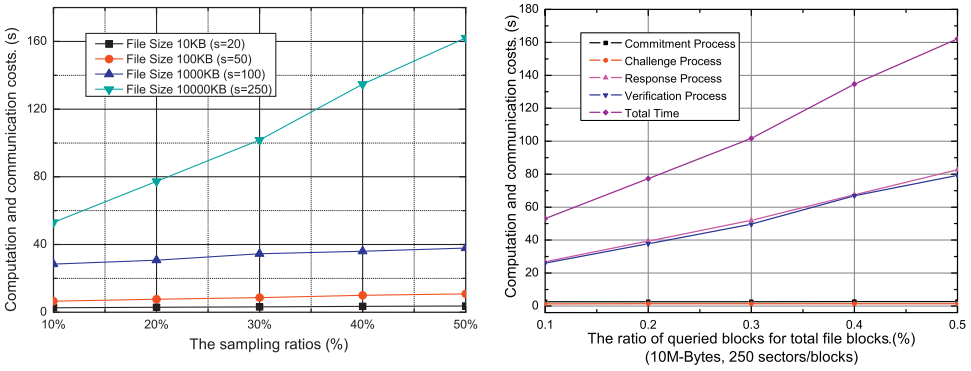

Fig. 9. Experiment results under different file size, sampling ratio, and sector number.

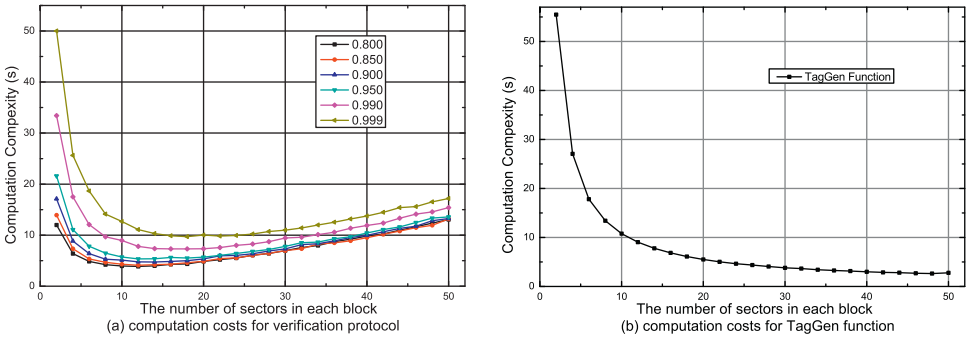

Fig. 10. Experiment results of different s for a 150 kB file ($\rho = 0.01$ and $P = 0.99$).

the servers can deceive the owners using previous metadata or responses due to lack of the randomness in the challenge. The number of updates and challenges is limited and fixed a priori. Also, one cannot perform block insertions anywhere. Based on this work, Erway et al. (2009) introduced two dynamic PDP schemes with a Hash function tree to realize $O(\log n)$ communication and computational costs for a file consisting of n blocks. The basic scheme, called DPDP-I, remains the drawback of SPDP, and in the 'blockless' scheme, called DPDP-II, the data blocks can be leaked by the response of challenge.

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

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

8. Conclusions

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

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

Acknowledgements

We thank the anonymous reviewers for their useful comments on this paper. The work of Yan Zhu was supported by the National Natural Science Foundation of China (Project No. 61170264 and No. 10990011). Gail-Joon Ahn and Hongxin Hu were partially supported by the Grants from US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308). This work of Stephen S. Yau was partially supported by the Grants from US National Science Foundation (NSF-CCF-0725340).

Appendix A. Security proof of construction

Proof. For some unavailable tags $\{\sigma^*\} \notin TagGen(sk, F)$, we assume that there exists an interactive machine P^*[3] can pass verification with noticeable probability, that is, there exists a polynomial $p(\cdot)$ and all sufficiently large κ's,

$$Pr[\langle P^*(F, \{\sigma^*\}), V\rangle(pk, \psi) = 1] \geq 1/p(\kappa). \tag{A.1}$$

Using P^*, we build a probabilistic algorithm $\mathcal{M}$ (called knowledge extractor) that breaks the computation Diffie–Hellman (*CDH*) problem in a cyclic group $\mathbb{G} \in \mathbb{S}$ of order p. That is, given $G, G_1, G_2 \in_R \mathbb{G}$, output $G^{ab} \in \mathbb{G}$, where $G_1 = G^a$, $G_2 = G^b$. The algorithm $\mathcal{M}$ is constructed by interacting with P^* as follows:

$$\begin{aligned}
e(\sigma'', h) &= e\left(\prod_{i\in I_1}(\xi^{(2)}_{t_i})^{v_i}, H''_1\right)\cdot e\left(\prod_{j=1}^{s} u_j^{\mu''_j}, H_2\right)\cdot(\pi'')^{-1} \\
&= e\left(\prod_{i\in I_1}(G^{r_i})^{v_i}, H''_1\right)\cdot\prod_{i\in I_2}(G^{r_i}\cdot G_2^{r'_i})^{v_i}\cdot e\left(\prod_{j=1}^{s} u_j^{\mu''_j}, H_2\right)\cdot\pi''^{-1} \\
&= e\left(\prod_{i\in I} G^{r_i\cdot v_i}, H'_1\right)\cdot e\left(\prod_{i\in I_2} G_2^{r'_i\cdot v_i}, H'_1\right) \\
&\quad\cdot\left(e\left(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2\right)\cdot\pi'^{-1}\right)^{\phi} \\
&= e\left(\prod_{i\in I} G^{r_i\cdot v_i}, H'_1\right)\cdot e\left(\prod_{i\in I_2} G_2^{r'_i\cdot v_i}, H'_1\right) \\
&\quad\cdot\left(e(\sigma', h)^{\phi}\cdot e\left(\prod_{i\in I} G^{r_i\cdot v_i}, H'_1\right)^{-\phi}\right) \\
&= e(\sigma', h)^{\phi}\cdot e\left(G_2^{\sum_{i\in I_2} r'_i v_i}\cdot G^{(1-\phi)\sum_{i\in I} r_i v_i}, H'_1\right).
\end{aligned} \tag{A.2}$$

Setup: $\mathcal{M}$ chooses a random $r \in_R \mathbb{Z}_p$ and sets $g = G$, $h = G^r$, $H_1 = G_1^r$, $H_2 = G_2^r$ as the public key $pk = (g, h, H_1, H_2)$, which is sent to P^*;

Learning: given a file $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$, $\mathcal{M}$ first chooses s random $\tau_i \in_R \mathbb{Z}_p$ and $u_i = G_2^{\tau_i}$ for $i \in [1, s]$. Secondly, $\mathcal{M}$ assigns the indexes $1, \ldots, n$ into two sets $T = \{t_1, \ldots, t_{n/2}\}$ and $T' = \{t'_1, \ldots, t'_{n/2}\}$. Let $m_{t_i,j} \neq m_{t'_i,j}$ for all $i \in [1, n/2]$ and $j \in [1, s]$. Then, $\mathcal{M}$ builds an index table χ and $\xi^{(1)}$ in terms of the original scheme and generates the tag of each block, as follows:

- For each $t_i \in T$, $\mathcal{M}$ chooses $r_i \in_R \mathbb{Z}_p$ and sets $\xi^{(2)}_{t_i} = H_{\xi^{(1)}}(\chi_{t_i}) = G^{r_i}$ and $\sigma_{t_i} = G_1^{r_i}\cdot G_2^{\sum_{j=1}^{s}\tau_j\cdot m_{t_i,j}}$.
- For each $t'_i \in T'$, $\mathcal{M}$ uses r_i and two random $r'_i, \zeta_i \in_R \mathbb{Z}_p$ to sets $\xi^{(2)}_{t'_i} = H_{\xi^{(1)}}(\chi_{t'_i}) = G^{r_i}\cdot G_2^{r'_i}$ and $\sigma_{t'_i} = G_1^{\zeta_i}\cdot G_2^{\sum_{j=1}^{s}\tau_j\cdot m_{t'_i,j}}$.

  $\mathcal{M}$ checks whether $e(\sigma_{t'_i}, h) \stackrel{?}{=} e(\xi^{(2)}_{t'_i}, H_1)\cdot e(\prod_{j=1}^{s} u_j^{m_{t'_i,j}}, H_2)$ for all $t'_i \in T'$. If the result is true, then outputs $G^{ab} = G_2^a = (G^{\zeta_i}\cdot G_1^{r_i})^{(r'_i)^{-1}}$, otherwise $\mathcal{M}$ sends $(F, \sigma^* = \{\sigma_i\}_{i=1}^n)$ and $\psi = (\xi^{(1)}, u = \{u_i\}, \chi)$ to P^*. At any time, P^* can query the hash function $H_{\xi^{(1)}}(\chi_k)$, $\mathcal{M}$ responds with $\xi^{(2)}_{t_i}$ or $\xi^{(2)}_{t'_i}$ with consistency, where $k = t_i$ or t'_i.

Output: $\mathcal{M}$ chooses an index set $I \subset [1, n/2]$ and two subset I_1 and I_2, where $I = I_1 \bigcup I_2$, $|I_2| > 0$. $\mathcal{M}$ constructs the challenges $\{v_i\}_{i\in I}$ and all $v_i \neq 0$. Then $\mathcal{M}$ simulates V to run an interactive $\langle P^*, \mathcal{M}\rangle$ as follows:

- *Commitment.* $\mathcal{M}$ receives (H'_1, π') from P^*;
- *Challenge.* $\mathcal{M}$ sends the challenge $Q_1 = \{(t_i, v_i)\}_{i\in I}$ to P^*;
- *Response.* $\mathcal{M}$ receives $(\sigma', \{\mu'_j\}_{j=1}^s)$ from P^*.

  $\mathcal{M}$ checks whether or not these responses is an effective result according to the verification equation in protocol. If it is true, then $\mathcal{M}$ completes a *rewindable* access to the prover P^* as follows:
- *Commitment.* $\mathcal{M}$ receives (H''_1, π'') from P^*;
- *Challenge.* $\mathcal{M}$ sends the following challenge to P^*, $Q_2 = \{(t_i, v_i)\}_{i\in I_1} \bigcup \{(t'_i, v_i)\}_{i\in I_2}$;
- *Response.* $\mathcal{M}$ receives $(\sigma'', \{\mu''_j\}_{j=1}^s)$ or a special halting-symbol from P^*.

  If the response is not a halting-symbol, then $\mathcal{M}$ checks whether the response is effective by Eq. (A.3), $H'_1 \stackrel{?}{=} H''_1$, and $\pi' \stackrel{?}{=} \pi''$. If they are true, then $\mathcal{M}$ computes

$$\gamma = \frac{\mu''_j - \mu'_j}{\sum_{i\in I_2} v_i\cdot(m_{t'_i,j} - m_{t_i,j})}$$

  for any $j \in [1, s]$ and verifies $H'_1 \stackrel{?}{=} H_1^{\gamma}$ to ensure this is an effective rewindable access. Finally, $\mathcal{M}$ outputs

$$G^{ab} = \left(\sigma''\cdot\sigma'^{-\phi}\cdot G_1^{\gamma\cdot(\phi-1)\sum_{i\in I} r_i v_i}\right)^{\frac{1}{\gamma\cdot\sum_{i\in I_2} r'_i\cdot v_i}}, \tag{A.3}$$

  where

$$\phi = \frac{\sum_{i\in I_1}\sum_{j=1}^{s}\tau_j m_{t_i,j} v_i + \sum_{i\in I_2}\sum_{j=1}^{s}\tau_j m_{t'_i,j} v_i}{\sum_{i\in I}\sum_{j=1}^{s}\tau_j m_{t_i,j} v_i}$$

  and $\phi \neq 1$.

It is obvious that we set $\alpha = a$ and $\beta = b$ in the above construction. Since the tags σ_{t_i} are available in $\forall t_i \in T$, the responses in the first interactive satisfies the equation:

$$\pi'\cdot e(\sigma', h) = e(\prod_{i\in I}(\xi^{(2)}_{t_i})^{v_i}, H'_1)\cdot e(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2) =$$

$$= e(G^{\sum_{i\in I} r_i\cdot v_i}, H'_1)\cdot e(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2).$$

[3] In this appendix, CSP and TPA are written as *P* and *V* for short, respectively.

However, the $\sigma_{t_i'}$ are unavailable in $\forall t_i' \in T'$. In the second interaction, we require that $\mathcal{M}$ can rewind the prover P^*, i.e., the chosen parameters are the same in two protocol executions (called as rewindable black-box knowledge extractor (Cramer et al., 2000; Goldreich, 2001)). In the above construction, this property ensures $H_1' = H_1''$, $\pi' = \pi''$, and for all $i \in [1, s]$,

$$\mu_j'' - \mu_j' = \gamma \cdot \sum_{i \in I} v_i \cdot (m_{t_i',j} - m_{t_i,j}) = \gamma \cdot \sum_{i \in I_2} v_i \cdot (m_{t_i',j} - m_{t_i,j}).$$

By checking $H_1' = H_1^{\gamma}$ for all γ computed from this equation, we can make sure of the consistence of $\lambda_i' = \lambda_i''$ for $i \in [1, s]$ in two executions. Thus, we have $e\left(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2\right) \cdot \pi' - 1 = e(G_2, H_2)^{\sum_{i \in I} \sum_{j=1}^{s} \tau_j m_{t_i,j} v_i}$ and

$$e(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2) \cdot \overline{\pi''}^{1} = e(G_2, H_2)^{\sum\sum_{i \in I_1} \sum_{j=1}^{s} \tau_j m_{t_i,j} v_i} \cdot e(G_2, H_2)^{\sum\sum_{i \in I_2} \sum_{j=1}^{s} \tau_j m_{t_i',j} v_i}.$$

This means that $e\left(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2\right) \cdot \overline{\pi''}^{1} = (e\left(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2\right) \cdot \overline{\pi'}^{1})^{\phi}$. In terms of the responses, we hold the Eq. (A.2). Hence, we have the equation $e(\sigma'' \cdot \overline{\sigma'}^{\phi}, h) = e\left(G_2^{\sum_{i \in I_2} r_i' \cdot v_i} \cdot G^{(1-\phi)\sum_{i \in I} r_i v_i}, H_1'\right)$, $H_1' = h^{a\gamma}$, and $G_1 = G^a$, thus the Eq. (A.3) holds. Furthermore, we have

$$Pr[\mathcal{M}(CDH(G, G^a, G^b)) = G^{ab}] \geq Pr[\langle P^*(F, \{\sigma^*\}), \mathcal{M}\rangle(pk, \psi) = 1] \geq 1/p(\kappa).$$

It follows that $\mathcal{M}$ can solve the given ϵ-CDH challenge with advantage at least ϵ, as required. This completes the proof of theorem. □

Appendix B. Proof of zero-knowledge

Proof. For the protocol $\mathcal{S}$, we construct a machine S, which is called a simulator for the interaction of V with P. Given the public key $pk = (g, h, H_1, H_2)$, for a file F, a public verification information $\psi = (\xi^{(1)}, u_1, \ldots, u_s, \chi)$, and an index set I ($t = |I|$), the simulator $S^*(pk, \psi)$ executes as follows:

1. Chooses a random $\sigma' \in_R \mathbb{G}$ and computes $e(\sigma', h)$;
2. Chooses t random coefficients $\{v_i\}_{i \in I} \in_R \mathbb{Z}_p^t$ and a random $\gamma \in_R \mathbb{Z}_p$ to compute $H_1' \leftarrow H_1^{\gamma}$ and $A_1 \leftarrow e(\prod_{i \in I} H_{\xi^{(1)}}(\chi_i)^{v_i}, H_1')$;
3. Chooses s random $\{\mu_i\} \in_R \mathbb{Z}_p^s$ to $A_2 \leftarrow e\left(\prod_{j=1}^{s} u_j^{\mu_j}, H_2\right)$;
4. Calculates $\pi \leftarrow A_1 \cdot A_2 \cdot e(\sigma', h)^{-1}$.
5. Outputs $(C, Q, \theta) = ((H_1', \pi), \{(i, v_i)\}_{i=1}^{t}, (\sigma', \mu))$ as the simulation results.

It is obvious that the output of simulator $S^*(pk, \psi)$ is an available verification for Eq. (A.3). Let $View(\langle P(F, \sigma), V^*\rangle(pk, \psi)) = ((\overline{H_1'}, \overline{\pi}), \{(i, \overline{v_i})_{i=1}^{t}\}, (\overline{\sigma'}, \overline{\mu}))$ denote the output of the interactive machine V^* after interacting with the interactive machine P on common input (pk, ψ). In fact, every pair of variables is identically distributed in two ensembles, for example, $\overline{H_1'}, \{(i, \overline{v_i})\}$ and $H_1', \{(i, v_i)\}$ are identically distributed due to $\gamma, \{v_i\} \in_R \mathbb{Z}_p$, as well as $(\overline{\sigma'}, \overline{\mu})$ and (σ', μ) is identically distributed due to $\sigma' \in_R \mathbb{G}, \lambda_j \in_R \mathbb{Z}_p$ and $u_j \leftarrow \lambda_j + \gamma \sum_{i \in I} v_i \cdot m_{i,j}$ for $i \in [1, s]$. Two variables, $\overline{\pi}$ and π, are computational distinguishable because the $\overline{\pi}$ is identically distributed in terms of the random choice of all λ_i and the distribution of π is decided on the randomized assignment of above variables.

Hence, the ensembles $S^*(pk, \psi)$ and $View(\langle P(F, \sigma), V^*\rangle(pk, \psi))$ is computationally indistinguishable, thus for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently large κ, it holds that

$$\left| Pr[D\left(pk, \psi, S^*(pk, \psi)\right) = 1] - Pr[D\left(pk, \psi, View(P(F, \sigma), V^*)(pk, \psi)\right)] = 1 \right| \leq \frac{1}{p(\kappa)}.$$

The fact that such simulators exist means that V^* does not gain any knowledge from P since the same output could be generated without any access to P. That is, the $\mathcal{S}$ is a zero-knowledge protocol. □

References

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M., 2010. A view of cloud computing. Commun. ACM 53 (4), 50–58.

Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X., 2007. Provable data possession at untrusted stores. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 598–609.

Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G., 2008. Scalable and efficient provable data possession. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, SecureComm, pp. 1–10.

Barreto, P.S.L.M., Galbraith, S.D., O'Eigeartaigh, C., Scott, M., 2007. Efficient pairing computation on supersingular abelian varieties. Des. Codes Cryptogr. 42 (3), 239–271.

Beuchat, J.-L., Brisebarre, N., Detrey, J., Okamoto, E., 2007. Arithmetic operators for pairing-based cryptography. In: Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, pp. 239–255.

Boneh, D., Boyen, X., Shacham, H.,2004. Short group signatures. In: In Proceedings of CRYPTO 04, LNCS Series. Springer-Verlag, pp. 41–55.

Boneh, D., Franklin, M., 2001. Identity-based encryption from the weil pairing. In: Advances in Cryptology (CRYPTO'2001). Vol. 2139 of LNCS, pp. 213–229.

Bowers, K.D., Juels, A., Oprea, A., 2009. Hail: a high-availability and integrity layer for cloud storage. In: ACM Conference on Computer and Communications Security, pp. 187–198.

Cramer, R., Damgård, I., MacKenzie, P.D., 2000. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Public Key Cryptography, pp. 354–373.

Dodis, Y., Vadhan, S.P., Wichs, D., 2009. Proofs of retrievability via hardness amplification. In: Reingold, O. (Ed.), Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009. Vol. 5444 of Lecture Notes in Computer Science. Springer, pp. 109–127.

Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R., 2009. Dynamic provable data possession. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, pp. 213–222.

Fu, K., Kaashoek, M.F., Mazières, D., 2002. Fast and secure distributed read-only file system. ACM Trans. Comput. Syst. 20 (1), 1–24.

Goldreich, O., 2001. Foundations of Cryptography: Basic Tools. Vol. Basic Tools. Cambridge University Press.

Hsiao, H.-C., Lin, Y.-H., Studer, A., Studer, C., Wang, K.-H., Kikuchi, H., Perrig, A., Sun, H.-M., Yang, B.-Y., 2009. A study of user-friendly hash comparison schemes. In: ACSAC, pp. 105–114.

Hu, H., Hu, L., Feng, D., 2007. On a class of pseudorandom sequences from elliptic curves over finite fields. IEEE Trans. Inform. Theory 53 (7), 2598–2605.

Juels Jr., A., Kaliski, B.S., 2007. Pors: proofs of retrievability for large files. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 584–597.

Ko, R.K.L., Lee, B.S., Pearson, S., 2011. Towards achieving accountability, auditability and trust in cloud computing. In: Abraham, A., Mauri, J.L., Buford, J.F., Suzuki, J., Thampi, S.M. (Eds.), Advances in Computing and Communications. Vol. 193 of Communications in Computer and Information Science. Springer, Berlin/Heidelberg, pp. 432–444.

Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L., 2006. Dynamic authenticated index structures for outsourced databases. In: Chaudhuri, S., Hristidis, V., Polyzotis, N. (Eds.), SIGMOD Conference. ACM, pp. 121–132.

Ma, D., Deng, R.H., Pang, H., Zhou, J., 2005. Authenticating query results in data publishing. In: Qing, S., Mao, W., Lopez, J., Wang, G. (Eds.), ICICS. Vol. 3783 of Lecture Notes in Computer Science. Springer, pp. 376–388.

Schnorr, C.-P., 1991. Efficient signature generation by smart cards. J. Cryptol. 4 (3), 161–174.

Shacham, H., Waters, B., 2008. Compact proofs of retrievability. In: Advances in Cryptology – ASIACRY, 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, pp. 90–107.

Tchifilionova, V., 2011. Security and privacy implications of cloud computing c lost in the cloud. In: Camenisch, J., Kisimov, V., Dubovitskaya, M. (Eds.), Open Research

Problems in Network Security. Vol. 6555 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 149–158.
Wang, C., Wang, Q., Ren, K., Lou, W., 2010. Privacy-preserving public auditing for data storage security in cloud computing. In: INFOCOM, 2010 Proceedings IEEE, pp. 1–9, 14-19.
Wang, Q., Wang, C., Li, J., Ren, K., Lou, W., 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of the 14th European Symposium on Research in Computer Security, ESORICS 2009, pp. 355–370.
Xie, M., Wang, H., Yin, J., Meng, X., 2007. Integrity auditing of outsourced data. In: Koch, C., Gehrke, J., Garofalakis, M.N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C.Y., Ganti, V., Kanne, C.-C., Klas, W., Neuhold, E.J. (Eds.), VLDB. ACM, pp. 782–793.
Yavuz, A.A., Ning, P., 2009. Baf: An efficient publicly verifiable secure audit logging scheme for distributed systems. In: ACSAC, pp. 219–228.
Yumerefendi, A.R., Chase, J.S., 2007. Strong accountability for network storage. ACM Trans. Storage (TOS) 3 (3).

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, Tempe. 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 in social network and cloud computing, network and distributed system security and secure software engineering.

Gail-Joon Ahn received the Ph.D. degree in information technology from George Mason University, Fairfax, VA, in 2000. 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 is currently an Associate Professor in the School of Computing, Informatics, and Decision Systems Engineering, Ira A. Fulton School of Engineering and the Director of Security Engineering for Future Computing Laboratory, Arizona State University, Tempe. 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.

Stephen S. Yau received the B.S. degree from National Taiwan University, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana, all in electrical engineering. He is Professor of Computer Science and Engineering and the Director of Information Assurance Center at Arizona State University, Tempe. He was previously with the University of Florida, Gainesville and Northwestern University, Evanston, Illinois. He has served as the president of the IEEE Computer Society and the Editor-in-Chief of Computer. His current research is in cyber security, distributed computing systems, software engineering, service-based computing and cloud computing systems. Contact him at yau@asu.edu or visit his Website: http://dpse.asu.edu/yau/.