

Dynamic Audit Services for Outsourced Storages in Clouds

Yan Zhu, *Member, IEEE*, Gail-Joon Ahn, *Senior Member, IEEE*,
Hongxin Hu, *Member, IEEE*, Stephen S. Yau, *Fellow, IEEE*,
Ho G. An, and Chang-Jun Hu

Abstract—In this paper, we propose a dynamic audit service for verifying the integrity of an untrusted and outsourced storage. Our audit service is constructed based on the techniques, fragment structure, random sampling, and index-hash table, supporting provable updates to outsourced data and timely anomaly detection. In addition, we propose a method based on probabilistic query and periodic verification for improving the performance of audit services. Our experimental results not only validate the effectiveness of our approaches, but also show our audit system verifies the integrity with lower computation overhead and requiring less extra storage for audit metadata.

Index Terms—Storage security, provable data possession, audit service, cloud storage

1 INTRODUCTION

CLOUD computing provides a scalable environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. Especially, the outsourced storage in clouds has 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 for storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to the clients because their data or archives are stored in an uncertain storage pool outside the enterprises. These security risks come from the following reasons: First, the cloud infrastructures are much more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machine) and external threats (e.g., via system holes) that can damage data integrity [1]; second, for the benefits of possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users [2]; furthermore, disputes occasionally suffer from the lack of trust on CSP because the data change may not be timely known by the cloud users, even if these disputes may result from the users' own improper operations [3]. Therefore, it is

necessary for CSP to offer an efficient audit service to check the integrity and availability of stored data [4].

Security audit is an important solution enabling trace-back and analysis of any activities including data accesses, security breaches, application activities, and so on. Data security tracking is crucial for all organizations that should comply with a wide range of federal regulations including the Sarbanes-Oxley Act, Basel II, HIPAA, and so on.¹ Furthermore, compared to the common audit, the audit services for cloud storages should provide clients with a more efficient proof for verifying the integrity of stored data. Unfortunately, the traditional cryptographic technologies, based on hash functions and signature schemes, cannot support for data integrity verification without a local copy of data. In addition, it is evidently impractical for audit services to download the whole data for checking data validation due to the communication cost, especially for large-size files. Therefore, following security and performance objectives should be addressed to achieve an efficient audit for outsourced storage in clouds:

- Y. Zhu and C.-J. Hu are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, 30 Xueyuan Road, Haidian District, Beijing 100083, China. E-mail: zhuyan@ustb.edu.cn, huchangjun@ies.ustb.edu.cn.
- G.-J. Ah and S.S. Yau are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, 699 S. Mill Avenue, Tempe, AZ 85281. E-mail: {gahn, yau}@asu.edu.
- H. Hu is with the Department of Computer and Information Sciences, Delaware State University, Dover, DE 19901. E-mail: hhu@desu.edu.
- H.G. An is with Google Inc. E-mail: hogeun78@gmail.com.

Manuscript received 28 Jan. 2011; revised 6 July 2011; accepted 21 July 2011; published online 18 Aug. 2011.

For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2011-01-0007. Digital Object Identifier no. 10.1109/TSC.2011.51.

- *Public auditability.* To allow a third party auditor (TPA) or clients with the help of TPA to verify the correctness of cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to cloud services;
- *Dynamic operations.* To ensure there is no attack to compromise the security of verification protocol or cryptosystem by using dynamic data operations;
- *Timely detection.* To detect data errors or losses in outsourced storage, as well as anomalous behaviors of data operations in a timely manner;
- *Effective forensic.* To allow TPA to exercise strict audit and supervision for outsourced data, and offer efficient evidences for anomalies; and

1. Source: <http://www.hhs.gov/ocr/privacy/>.

TABLE 1
Comparison of POR/PDP Schemes for a File Consisting of n Blocks

Scheme	CSP comp.	Client Comp.	Comm.	Frag.	Privacy	Dynamic Operations			Prob. of Detection
						modify	insert	delete	
PDP[5]	$O(t)$	$O(t)$	$O(1)$		✓				$1 - (1 - \rho)^t$
SPDP[6]	$O(t)$	$O(t)$	$O(t)$	✓	✓	✓ [#]		✓ [#]	$1 - (1 - \rho)^{t \cdot s}$
DPDP-I[7]	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$		✓	✓	✓	✓	$1 - (1 - \rho)^t$
DPDP-II[7]	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$			✓	✓	✓	$1 - (1 - \rho)^{\Omega(n)}$
CPOR-I[8]	$O(t)$	$O(t)$	$O(1)$						$1 - (1 - \rho)^t$
CPOR-II[8]	$O(t + s)$	$O(t + s)$	$O(s)$	✓					$1 - (1 - \rho)^{t \cdot s}$
Our Scheme	$O(t + s)$	$O(t + s)$	$O(s)$	✓	✓	✓	✓	✓	$1 - (1 - \rho)^{t \cdot s}$

s is the number of sectors in each block, t is the number of sampling blocks, [#] indicates that an operation is performed with a limited (pre-determined) number of times, ρ is the probability of block corruption in a cloud server.

- *Lightweight.* To allow TPA to perform audit tasks with the minimum storage, lower communication cost, and less computation overhead.

In this paper, we introduce a dynamic audit service for integrity verification of untrusted and outsourced storages. Constructed on interactive proof system (IPS) with the zero-knowledge property, our audit service can provide public auditability without downloading raw data and protect privacy of the data. Also, our audit system can support dynamic data operations and timely anomaly detection with the help of several effective techniques, such as fragment structure, random sampling, and index-hash table (IHT). We also propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services. A proof-of-concept prototype is also implemented to evaluate the feasibility and viability of our proposed approaches. Our experimental results not only validate the effectiveness of our approaches, but also show that our system does not create any significant computation cost and require less extra storage for integrity verification.

We list the features of our scheme in Table 1. We also make a comparison of related techniques, involving provable data possession (PDP) [5], scalable PDP (SPDP) [6], dynamic PDP (DPDP) [7], and compact proofs of retrievability (CPOR) [8]. It clearly shows that our scheme not only supports complete privacy protection and dynamic data operations, but also enables significant savings in computation and communication costs, as well as a high-detection probability of disrupted blocks.

The rest of the paper is organized as follows: Section 2 describes the research background and related work. Section 3 addresses our audit system architecture and main techniques. Sections 4 and 5 describe the definition and construction of corresponding algorithms, respectively. In Sections 6, we present the security of our schemes along with the performance of experimental results in Section 7. Finally, we conclude this paper in Section 8.

2 BACKGROUND AND RELATED WORK

Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature schemes [9], [10], [11], 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 communications, especially for large-size

files. Moreover, the ability to audit the correctness of data in a cloud environment can be formidable and expensive for cloud users. Therefore, it is crucial to realize public auditability for CSS, so that data owners (DOs) may resort to a 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.

To implement public auditability, the notions of proof of retrievability (POR) [2] and PDP [5] have been proposed by some researchers. These approaches were based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact. 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, they help 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.

There exist some solutions for audit services on outsourced data. For example, Xie et al. [12] proposed an efficient method on content comparability for outsourced database, but it was not suitable for irregular data. Wang et al. [13] also provided a similar architecture for public audit services. To support their architecture, a public audit scheme was proposed with privacy-preserving property. However, the lack of rigorous performance analysis for a constructed audit system greatly affects the practical application of their scheme. For instance, in this scheme an outsourced file is directly split into n blocks, and then each block generates a verification tag. To maintain security, the length of block must be equal to the size of cryptosystem, that is, 160 bits, which is 20 bytes. This means that 1M bytes file is split into 50,000 blocks and generates 50,000 tags [8], and the storage of tags is at least 1M bytes. Therefore, it is inefficient to build an audit system based on this scheme. To address such a problem, we introduce a fragment technique to improve the system performance and reduce the extra storage (see Section 3.1).

Another major concern is the security issue of dynamic data operations for public audit services. In clouds, one of the core design principles is to provide dynamic scalability for various applications. This means that remotely stored data might be not only accessed by the clients but also dynamically updated by them, for instance, through block operations such as modification, deletion and insertion.

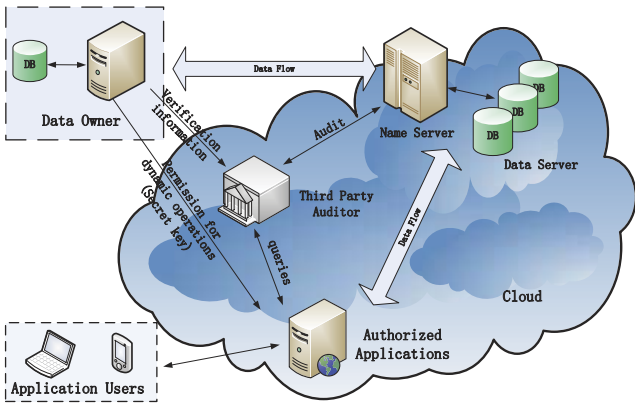


Fig. 1. Audit system architecture.

However, these operations may raise security issues in most of existing schemes, e.g., the forgery of the verification metadata (called as tags) generated by DOs and the leakage of the user’s secret key. Hence, it is crucial to develop a more efficient and secure mechanism for dynamic audit services, in which a potential adversary’s advantage through dynamic data operations should be prohibited.

3 ARCHITECTURE AND TECHNIQUES

We introduce an audit system architecture for outsourced data in clouds as shown in Fig. 1. In this architecture, we consider that a data storage service involves four entities: DO, who has a large amount of data to be stored in the cloud; CSP, who provides data storage service and has enough storage space and computation resources; TPA, who has capabilities to manage or monitor the outsourced data under the delegation of DO; and authorized applications (AAs), who have the right to access and manipulate the stored data. Finally, application users can enjoy various cloud application services via these AAs.

We assume the TPA is reliable and independent through the following audit functions: TPA should be able to make regular checks on the integrity and availability of the delegated data at appropriate intervals; TPA should be able to organize, manage, and maintain the outsourced data instead of DOs, and support dynamic data operations for AAs; and TPA should be able to take the evidences for disputes about the inconsistency of data in terms of authentic records for all data operations.

To realize these functions, our audit service is comprised of three processes:

- *Tag generation.* The client (DO) uses a secret key sk to preprocess a file, which consists of a collection of n blocks, generates a set of public verification parameters (PVPs) and IHT that are stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy (see Fig. 2a);
- *Periodic sampling audit.* By using an interactive proof protocol of retrievability, TPA (or other applications) issues a “random sampling” challenge to audit the integrity and availability of the outsourced data in terms of verification information (involving PVP and IHT) stored in TPA (see Fig. 2b); and
- *Audit for dynamic operations.* An AA, who holds a DO’s secret key sk , can manipulate the outsourced

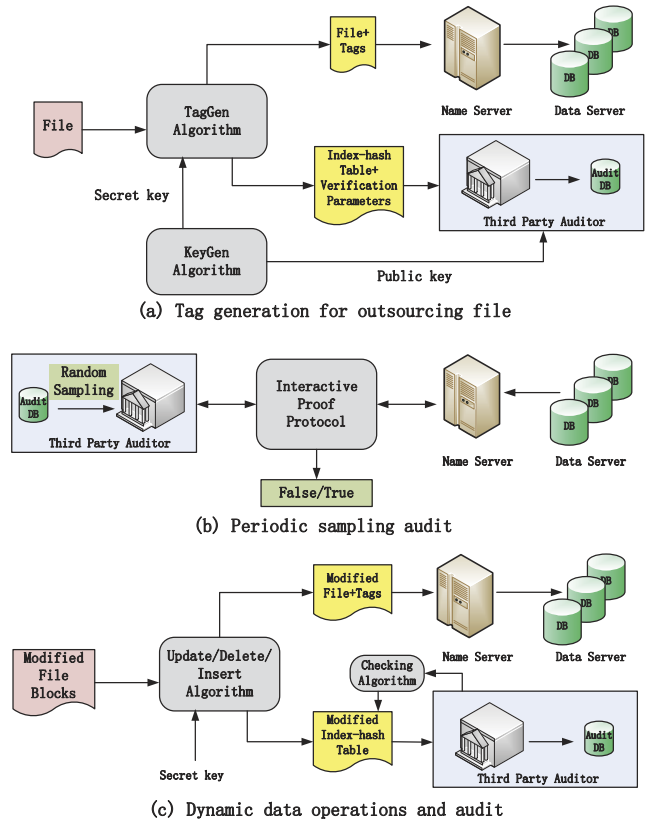


Fig. 2. Three processes of our audit system.

data and update the associated IHT stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the AAs and forge the valid audit records (see Fig. 2c).

In general, the AAs should be cloud application services inside clouds for various application purposes, but they must be specifically authorized by DOs for manipulating outsourced data. Since the acceptable operations require that the AAs must present authentication information for TPA, any unauthorized modifications for data will be detected in audit processes or verification processes. Based on this kind of strong authorization-verification mechanism, we assume neither CSP is trusted to guarantee the security of stored data, nor a DO has the capability to collect the evidence of CSP’s faults after errors have been found.

The ultimate goal of this audit infrastructure is to enhance the credibility of CSSs, but not to increase DO’s burden. Therefore, TPA should be constructed in clouds and maintained by a CSP. To ensure the trust and security, TPA must be secure enough to resist malicious attacks, and it should be strictly controlled to prevent unauthorized accesses even for internal members in clouds. A more practical way is that TPA in clouds should be mandated by a trusted third party (TTP). This mechanism not only improves the performance of an audit service, but also provides the DO with a maximum access transparency. This means that DOs are entitled to utilize the audit service without additional costs.

The aforementioned processes involve some procedures: *KeyGen*, *TagGen*, *Update*, *Delete*, *Insert* algorithms, as well as an *Interactive Proof Protocol* of Retrievability. We make use

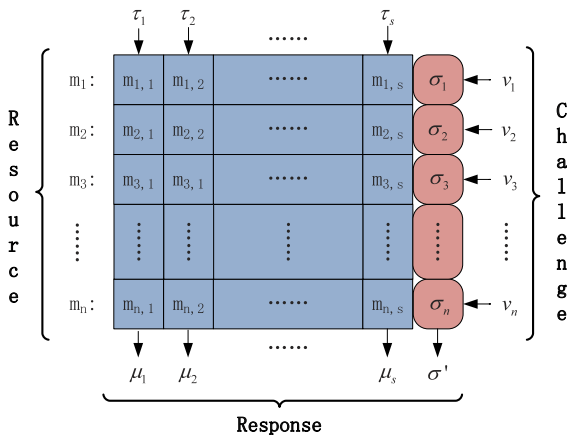


Fig. 3. Fragment structure and sampling audit.

of the following techniques to construct corresponding algorithms and protocols.

3.1 Fragment Structure and Secure Tags

To maximize the storage efficiency and audit performance, our audit system introduces a general fragment structure for outsourced storages. An instance for this framework, which is used in our approach is shown in Fig. 3: An outsourced file F is split into n blocks $\{m_1, m_2, \dots, m_n\}$, and each block m_i is split into s sectors $\{m_{i,1}, m_{i,2}, \dots, m_{i,s}\}$. The fragment framework consists of n block-tag pair (m_i, σ_i) , where σ_i is a signature tag of a block m_i generated by some secrets $\tau = (\tau_1, \tau_2, \dots, \tau_s)$. We can use such tags and corresponding data to construct a response in terms of the TPA's challenges in the verification protocol, such that this response can be verified without raw data. If a tag is unforgeable by anyone except the original signer, we call it a *secure tag*.

Finally, these block-tag pairs are stored in CSP and the encrypted secrets τ (called as PVP) are in TTP. Although this fragment structure is simple and straightforward, but the 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 with the increase of s . Hence, this structure can reduce the extra storage for tags and improve the audit performance.

There exist some schemes for the convergence of s blocks to generate a secure signature tag, e.g., MAC-based, ECC, or RSA schemes [5], [8]. These schemes, built from collision-resistance hash functions (see Section 5) and a random oracle model, support the features of scalability, performance, and security.

3.2 Periodic Sampling Audit

In contrast with "whole" checking, random "sampling" checking greatly reduces the workload of audit services, while still achieves an effective detection of misbehaviors. Thus, a probabilistic audit on sampling checking is preferable to realize the anomaly detection in a timely manner, as well as to rationally allocate resources. The fragment structure shown in Fig. 3 provides probabilistic audit as well: Given a randomly chosen challenge (or query) $Q = \{(i, v_i)\}_{i \in I}$, where I is a subset of the block indices and v_i is a random coefficient, an efficient algorithm is used to

TABLE 2
IHT with Random Values

No.	B_i	V_i	R_i	
0	0	0	0	← Used to head
1	1	2	r'_1	← Update
2	2	1	r_2	
3	4	1	r_3	← Delete
4	5	1	r_5	
5	5	2	r'_5	← Insert
\vdots	\vdots	\vdots	\vdots	
n	n	1	r_n	
$n+1$	$n+1$	1	r_{n+1}	← Append

produce a constant-size response $(\mu_1, \mu_2, \dots, \mu_s, \sigma')$, where μ_i comes from all $\{m_{k,i}, v_k\}_{k \in I}$ and σ' is from all $\{\sigma_k, v_k\}_{k \in I}$. Generally, this algorithm relies on homomorphic properties to aggregate data and tags into a constant-size response, which minimizes network communication costs.

Since the single sampling checking may overlook a small number of data abnormality, we propose a periodic sampling approach to audit outsourced data, which is named as *Periodic Sampling Audit*. With this approach, the audit activities are efficiently scheduled in an audit period, and a TPA merely needs to access small portions of files to perform audit in each activity. Therefore, this method can detect exceptions periodically, and reduce the sampling numbers in each audit.

3.3 Index-Hash Table

To support dynamic data operations, we introduce a simple IHT to record the changes of file blocks, as well as generate the hash value of each block in the verification process. The structure of our IHT is similar to that of file block allocation table in file systems. Generally, the IHT χ consists of serial number, block number, version number, and random integer (see Table 2 in Section 5). Note that we must assure all records in the IHT differ from one another to prevent the forgery of data blocks and tags. In addition to recording data changes, each record χ_i in the table is used to generate a unique hash value, which in turn is used for the construction of a signature tag σ_i by the secret key sk . The relationship between χ_i and σ_i must be cryptographically secure, and we make use of it to design our verification protocol.

Although the IHT may increase the complexity of an audit system, it provides a higher assurance to monitor the behavior of an untrusted CSP, as well as valuable evidence for computer forensics, due to the reason that anyone cannot forge the valid χ_i (in TPA) and σ_i (in CSP) without the secret key sk .

In practical applications, this architecture can be constructed into a virtualization infrastructure of cloud-based storage service [14]. In Fig. 4, we show an example of Hadoop distributed file system (HDFS),² which is a distributed, scalable, and portable file system [15]. HDFS'

2. Hadoop enables 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.

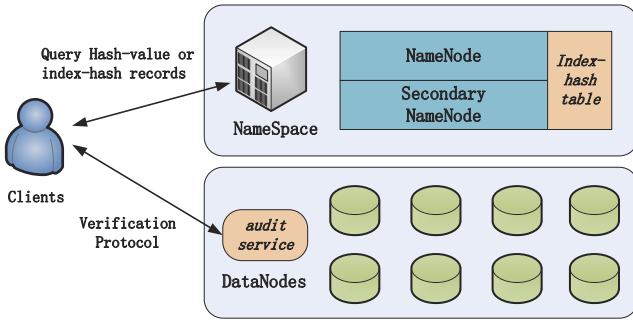


Fig. 4. An example of hash index hierarchy in HDFS.

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 dynamic audit, the IHT 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 these hash values, the clients or TPA can implement a verification protocol via audit services. Hence, it is easy to replace the common checksum algorithm with our scheme for anomaly detection without downloading data in current HDFS.

4 ALGORITHMS FOR AUDIT SYSTEM

In this section, we describe the construction of algorithms in our audit architecture. First, we present the definitions for the tag generation process as follows:

- $KeyGen(1^\kappa)$ takes a security parameter κ as an input, and returns a public/secret keypair (pk, sk) ; and
- $TagGen(sk, F)$ takes a secret key sk and a file F , and returns a triple (τ, ψ, σ) , where τ denotes the secret used to generate verification tags, ψ is a set of PVPs u and IHT χ , i.e., $\psi = (u, \chi)$, and σ denotes a set of tags.

A DO or AAs only need to save the secret key sk —that is, sk would not be necessary for the verification/audit process. The secret of the processed file τ can be discarded after tags are generated due to PVPs u .

Fig. 5 demonstrates the workflow of our audit system. Suppose a DO wants to store a file in a storage server, and maintains a corresponding authenticated index structure at a TPA. As shown in Fig. 5a, using $KeyGen()$, the owner first generates a public/secret keypair (pk, sk) by himself or the system manager, and sends his public key pk to TPA. Note that TPA cannot obtain the client’s secret key sk . Then, the owner chooses a random secret τ and invokes $TagGen()$ to produce public verification information $\psi = (u, \chi)$ and signature tags σ , where τ is unique for each file and χ is an IHT. Finally, the owner sends ψ and (F, σ) to TPA and CSP, respectively.

4.1 Supporting Periodic Sampling Audit

At any time, TPA can check the integrity of a file F as follows: TPA first queries database to obtain the verification information ψ and initializes an interactive protocol $Proof(CSP, Client)$; then, it performs a 3-move proof protocol: *Commitment*, *Challenge*, and *Response*; and it finally verifies the interactive data to get the results. In fact, because our scheme is a publicly verifiable protocol, anyone can run this protocol, but s/he is unable to get any advantage to break the cryptosystem, even if TPA and CSP cooperate for an attack. Let $P(x)$ 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 process can be defined as follows:

$Proof(CSP, TPA)$. is an interactive proof protocol between CSP and TPA, that is, $\langle CSP(F, \sigma), TPA \rangle(pk, \psi)$, where a public key pk and a set of public parameters ψ are the common inputs between TPA and CSP, and CSP takes a file F and a set of tags σ . At the end of the protocol, TPA returns $\{0|1\}$, where 1 means the file is correctly stored on the server. \square

An audit service executes the verification process periodically by using the above-mentioned protocol. Fig. 5b shows such a two-party protocol between TPA and CSP, i.e., $Proof(CSP, TPA)$, without the involvement of a client (DO or AA). It also shows two verification processes. To improve the efficiency of verification process, TPA performs audit tasks based on a probabilistic sampling.

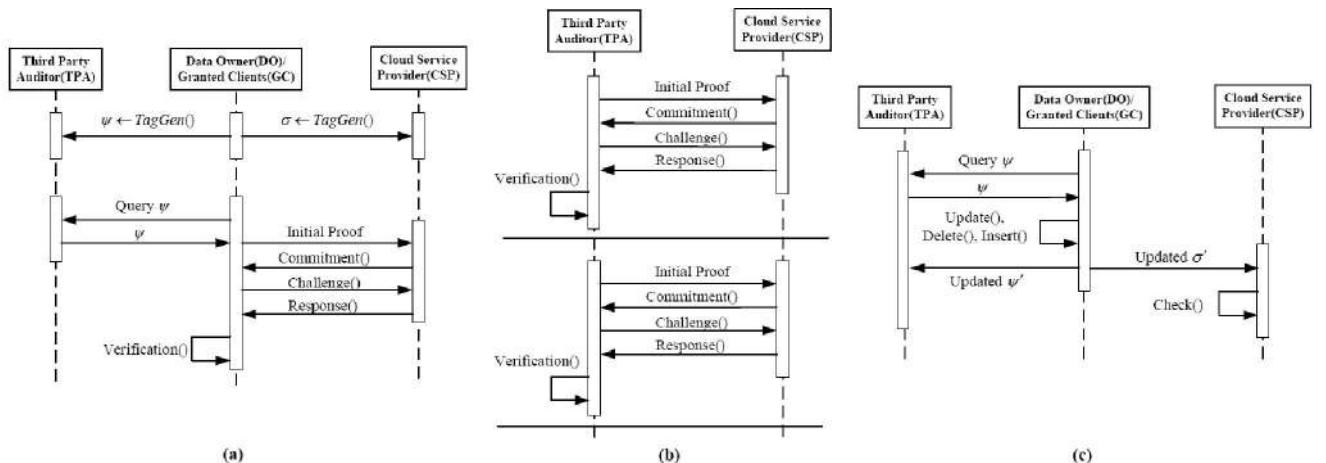


Fig. 5. Workflow of audit system: (a) tag generation and user’s verification, (b) periodic sampling audit, and (c) dynamic data operations and audit.

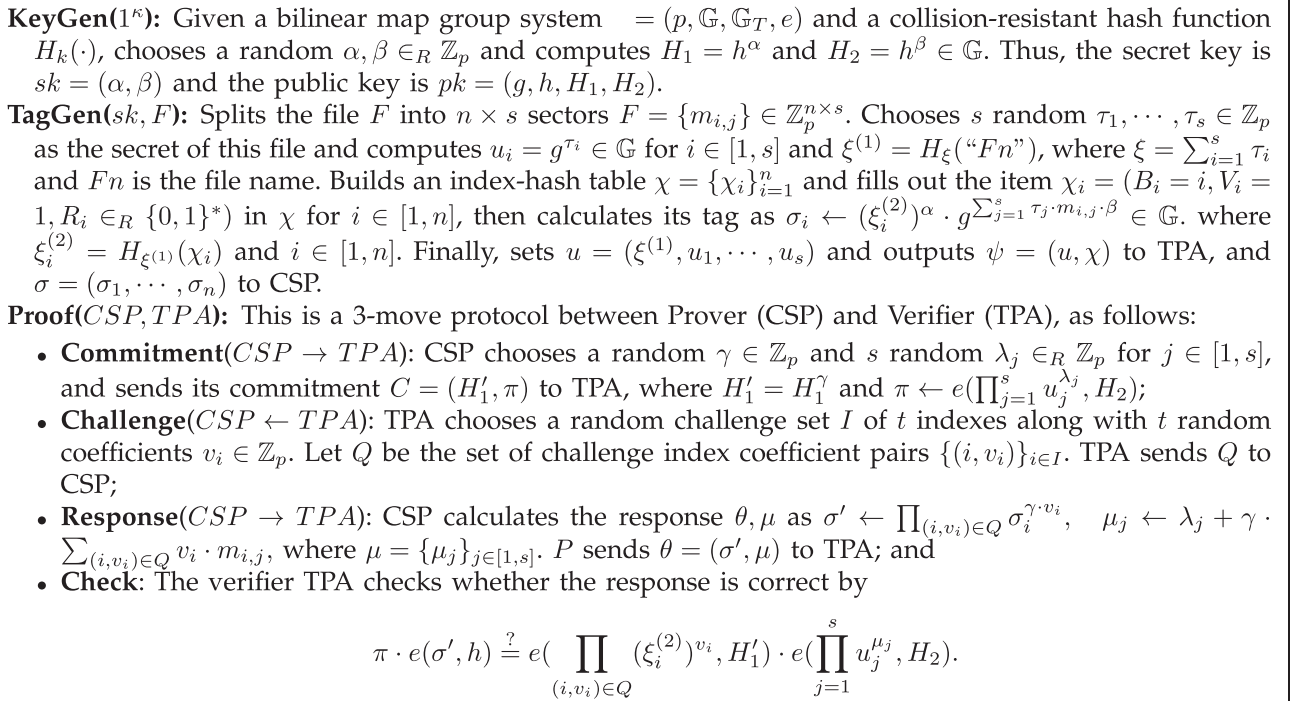


Fig. 6. Proposed IPOR scheme for key generation, tag generation, and verification protocol.

4.2 Supporting Dynamic Data Operations

To meet the requirements from dynamic scenarios, we introduce following definitions for dynamic data operations:

- *Update(sk, χ_i, m'_i)* is an algorithm run by AA to update the block of a file m'_i at the index i by using sk , and it returns a new verification metadata $(\chi'_i, \sigma'_i, m'_i)$;
- *Delete(sk, χ_i, m_i)* is an algorithm run by AA to delete the block m_i of a file m_i at the index i by using sk , and it returns a new verification metadata $(\chi'_i, \sigma'_i, m'_i)$; and
- *Insert(sk, χ_i, m_i)* is an algorithm run by AA to insert the block of a file m_i at the index i by using sk , and it returns a new verification metadata $(\chi'_i, \sigma'_i, m'_i)$.

To ensure the security, dynamic data operations are available only to DOs or AAs, who hold the secret key sk . Here, all operations are based on data blocks. Moreover, to implement audit services, applications need to update the IHTs. It is necessary for TPA and CSP to check the validity of updated data. In Fig. 5c, we describe the process of dynamic data operations and audit. First, an AA obtains the public verification information ψ from TPA. Second, the application invokes the *Update*, *Delete*, and *Insert* algorithms, and then sends the new ψ' and σ' to TPA and CSP, respectively. Next, the CSP makes use of an algorithm *Check* to verify the validity of updated data. Note that the *Check* algorithm is important to ensure the effectiveness of the audit. Finally, TPA modifies audit records after the confirmation message from CSP is received and the completeness of records is checked.

5 CONSTRUCTION FOR OUR SCHEME

We propose an efficient interactive POR (IPOR) scheme to realize the integer verification of outsourced data. This

scheme includes a 3-move interactive proof protocol, which also provides privacy protection property to ensure the confidentiality of secret data.

5.1 Notations and Preliminaries

Let $\mathcal{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0, 1\}^* \rightarrow \{0, 1\}^n$ indexed by $k \in \mathcal{K}$. We say that an 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 [16].

We set up our systems using bilinear pairings proposed by Boneh and Franklin [17]. Let \mathbb{G} and \mathbb{G}_T be two multiplicative groups using elliptic curve conventions with a large prime order p . The function e be a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties: for any $G, H \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$, we have: 1) Bilinearity: $e(G^a, H^b) = e(G, H)^{ab}$; 2) Nondegeneracy: $e(G, H) \neq 1$ unless G or $H = 1$; and 3) Computability: $e(G, H)$ is efficiently computable. A bilinear map system is a tuple $\mathbb{S} = \langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ composed of the objects as described above.

5.2 Proposed Construction

We present our IPOR construction in Fig. 6. In our scheme, each client holds a secret key sk , which can be used to generate the tags of many files. Each processed file produces a PVP $\psi = (u, \chi)$, where $u = (\xi^{(1)}, u_1, \dots, u_s)$, $\chi = \{\chi_i\}_{i \in [1, n]}$ is the IHT. We define $\chi_i = (B_i \| V_i \| R_i)$, where B_i is a sequence number of block, V_i is a version number of updates for this block, and R_i is a random integer to avoid collision. The value $\xi^{(1)}$ can be considered as the signature of the secret τ_1, \dots, τ_s . Note that it must assure that ψ s are different for all processed files.

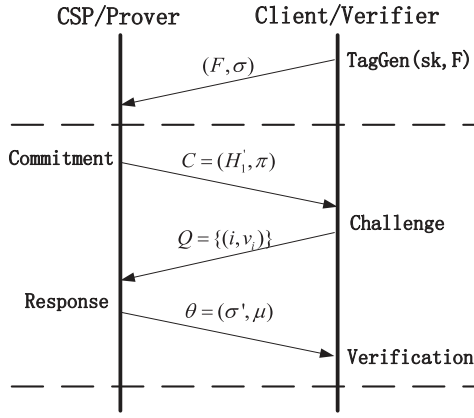


Fig. 7. Framework of a IPOR model.

In our construction, the verification protocol has a 3-move structure: commitment, challenge, and response, as shown in Fig. 7. This protocol is similar to Schnorr's Σ protocol [18], which is a zero-knowledge proof system. By using this property, we ensure the verification process does not reveal anything. To prevent the leakage of stored data and tags in the verification process, the private 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 prevent the adversaries from gaining those properties.

Moreover, it is obvious that this construction admits a short constant-size response $\theta = (\sigma', \mu) \in \mathbb{G} \times \mathbb{Z}_p^s$ without being dependent on the size of challenge. That is extremely important for large-size files.

5.3 Implementation of Dynamic Operations

To support dynamic data operations, it is necessary for TPA to employ an IHT χ to record the current status of the stored files. Some existing index schemes for dynamic scenarios are insecure due to replay attack on the same Hash values. To solve this problem, a simple IHT $\chi = \{\chi_i\}$ is used as described in Table 2, which includes four columns: No. denotes the real number i of data block m_i , B_i is the original number of block, V_i stores the version number of updates for this block, and R_i is a random integer to avoid collision.

To ensure the security, we require that each $\chi_i = "B_i||V_i||R_i"$ is unique in this table. Although the same values of $"B_i||V_i"$ may be produced by repeating the insert and delete operations, the random R_i can avoid this collision. An alternative method is to generate an updated random value by $R'_i \leftarrow H_{R_i}(\sum_{j=1}^s m'_{i,j})$, where the initial value is $R_i \leftarrow H_{\xi^{(1)}}(\sum_{j=1}^s m_{i,j})$ and $m_i = \{m_{i,j}\}$ denotes the i th data block. We show a simple example to describe the change of IHT for different operations in Table 2, where an empty record ($i = 0$) is used to support the operations on the first record. The "Insert" operation on the last record is replaced with "Append" operation. It is easy to prove that each χ_i is unique in χ in our scheme.

Based on the construction of IHTs, we propose a simple method to provide dynamic data modification as illustrated in Fig. 8. All tags and the IHT should be renewed and reorganized periodically to improve the performance. Obviously, we can replace the sequent lists with dynamically linked lists to improve the efficiency of updating the IHT.

Update(sk, ψ, m'_i): modifies the version number by $V_i \leftarrow \max_{B_i=B_j} \{V_j\} + 1$ and chooses a new R_i in $\chi_i \in \chi$ to get a new χ'_i ; computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}("B_i||V_i||R_i")$; by using sk , computes $\sigma'_i = (\xi_i^{(2)})^\alpha \cdot (\prod_{j=1}^s u_j^{m'_{i,j}})^\beta$, where $u = \{u_j\} \in \psi$, finally outputs $O = (\chi'_i, \sigma'_i, m'_i)$.

Delete(sk, ψ, m_i): computes the original σ_i by m_i and computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}("B_i||0||R_i")$ and $\sigma'_i = (\xi_i^{(2)})^\alpha$ by sk ; deletes i -th record to get a new ψ' ; finally outputs $O = (\chi'_i, \sigma_i, \sigma'_i)$.

Insert(sk, ψ, m'_i): inserts a new record in i -th position of the index-hash table $\chi \in \psi$, and the other records move backward in order; modifies $B_i \leftarrow B_{i-1}$, $V_i \leftarrow \max_{B_i=B_j} \{V_j\} + 1$, and a random R_i in $\chi_i \in \chi$ to get a new χ'_i ; computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}("B_i||V_i||R_i")$ and $\sigma'_i = (\xi_i^{(2)})^\alpha \cdot (\prod_{j=1}^s u_j^{m'_{i,j}})^\beta$, where $u = \{u_j\} \in \psi$, finally outputs $O = (\chi'_i, \sigma'_i, m'_i)$.

Check($U/D/I, O$): The application sends the operation type $U/D/I$ and the result O is given to CSP via a secure channel.

- For Update or Insert operations, CSP must check whether the following is held: for $(\chi'_i, \sigma'_i, m'_i)$, $e(\sigma'_i, h) \stackrel{?}{=} e(\xi_i^{(2)}, H_1) \cdot e(\prod_{j=1}^s u_j^{m'_{i,j}}, H_2)$; and
- For Delete operation, CSP must check whether σ_i is equal to the stored σ_i and $e(\sigma'_i, h) \stackrel{?}{=} e(H_{\xi^{(1)}}("B_i||0||R_i"), H_1)$.

In addition, TPA must replace χ_i with the new χ'_i and check the completeness of $\chi'_i \in \psi$.

Fig. 8. Algorithms for dynamic operations.

6 SECURITY ANALYSIS

First, we prove the completeness of our construction: For every available tag $\sigma \in TagGen(sk, F)$ and a random challenge $Q = (i, v_i)_{i \in I}$, the protocol always passes the verification test, that is, $\Pr[(CSP(F, \sigma), TPA^*)(pk, \psi) = 1] = 1$. We can prove this equation holds by using

$$\begin{aligned}
 \pi \cdot e(\sigma', h) &= e(g, h)^\beta \sum_{j=1}^s \tau_j \cdot \lambda_j \cdot e\left(\prod_{(i, v_i) \in Q} (\xi_i^{(2)})^{v_i}, h\right)^{\alpha \cdot \gamma} \\
 &\quad \cdot e(g, h)^{\gamma \cdot \beta \sum_{j=1}^s (\tau_j \cdot \sum_{(i, v_i) \in Q} v_i \cdot m_{i,j})} \\
 &= e(g, h)^\beta \sum_{j=1}^s \tau_j \cdot \lambda_j \cdot e\left(\prod_{(i, v_i) \in Q} (\xi_i^{(2)})^{v_i}, h\right)^{\alpha \cdot \gamma} \\
 &\quad \cdot e(g, h)^{\beta \sum_{j=1}^s (\tau_j \cdot \mu_j - \tau_j \cdot \lambda_j)} \\
 &= e\left(\prod_{(i, v_i) \in Q} (\xi_i^{(2)})^{v_i}, h^{\alpha \cdot \gamma}\right) \cdot \prod_{j=1}^s e(u_j^{\mu_j}, h^\beta).
 \end{aligned} \tag{1}$$

Next, to protect the confidentiality of checked data, we are more concerned about the leakage of private information

(which includes the verified data $\{m_i\}$ and their tags $\{\sigma_i\}$) in public verification process. To address this problem, we introduce Zero-Knowledge property into our construction:

Definition 1 (Zero-knowledge). An interactive POR scheme is computational zero knowledge if there exists a probabilistic polynomial-time algorithm S^* (call a Simulator) such that for every probabilistic polynomial-time (PPT) algorithm D and V^* , every polynomial $p(\cdot)$, and all sufficiently large s , it holds that

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

where, $S^*(pk, \psi)$ denotes the output of simulator S . That is, for all $\sigma \in \text{TagGen}(sk, F)$, the ensembles $S^*(pk, \psi)$ and $\langle P(F, \sigma), V^* \rangle(pk, \psi)^3$ are computationally indistinguishable.

Actually, zero-knowledge is a property that captures P 's robustness against attempts to gain knowledge by interacting with it. For our IPOR scheme, we make use of the zero-knowledge property to guarantee the security of data blocks and signature tags. We have the following theorem (see Appendix A, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TSC.2011.51>):

Theorem 1. Our IPOR scheme is a (computational) zero-knowledge PDP (called as ZKPOR) with respect to the polynomial-time simulators.

Then, we turn attention to the audit security for dynamic operations. It is easy to discover that the security of our scheme against dynamic operations is built on collision resistant of all hash values $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$, where $\xi^{(1)} = H_{\xi}(\text{"Fn"})$, $\xi = \sum_{i=1}^s \tau_i \pmod p$ and $\chi_i = \text{"}B_i \| V_i \| R_i'' \in \chi$. First, in an IHT $\chi = \{\chi_i\}$ and $\chi_i = \text{"}B_i \| V_i \| R_i''$, there exists no identical records χ_i and χ_j for all dynamic operations only if $B_i \neq B_j$, $V_i \neq V_j$, or $R_i \neq R_j$ for any indexes $i, j \in \mathbb{N}$. Furthermore, the secrets $\{\tau_1, \dots, \tau_s\} \in \mathbb{Z}_p^s$ are also used to avoid a collision of files that have the same file name. For both mechanisms, we can prove the following theorem (see Appendix B, available in the online supplemental material):

Theorem 2. The hash values $\xi_i^{(2)}$ is $(\varepsilon, \sqrt{2^{L+2}p} \ln \frac{1}{1-\varepsilon})$ collision-resistant in our scheme,⁴ even if a client generates $\sqrt{2p} \cdot \ln \frac{1}{1-\varepsilon}$ files with the same file 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 $|R_i| = L$.

Based on collision resistant of χ , we consider a new notion of security for our scheme on dynamic operations, which is called dynamic existential unforgeability under an adaptive chosen message attack [19], [20]. This kind of security can be defined using the following game between a challenger \mathcal{B} and an adversary \mathcal{A} :

3. The output of the interactive machine V^* after interacting with $P(F, \sigma)$ on common input (pk, ψ) .

4. We use the terminology (ε, Q) to denote an algorithm with average-case success probability ε , in which the number of oracle queries made by the algorithm is at most Q .

1. *Initial.* Given a file F , the challenger \mathcal{B} simulates $\text{KeyGen}(1^\kappa)$ and $\text{TagGen}(sk, F)$ to generate the public parameters pk and ψ , and sends them to the adversary \mathcal{A} .
2. *Learning.* \mathcal{A} adaptively issues at most q_t times queries q_1, \dots, q_t to learn the information of tags via dynamic operations, as follows:
 - a. *Update query* (i, m'_i) . \mathcal{B} generates $(\chi'_i, \sigma'_i, m'_i) \leftarrow \text{Update}(sk, \chi_i, m'_i)$ and sends it to \mathcal{A} ;
 - b. *Delete query* (i) . \mathcal{B} generates $(\chi'_i, \sigma_i, \sigma'_i) \leftarrow \text{Delete}(sk, \chi_i, m_i)$ and sends it to \mathcal{A} ;
 - c. *Insert query* (i, m'_i) . \mathcal{B} generates $(\chi'_i, \sigma'_i, m'_i) \leftarrow \text{Insert}(sk, \chi_i, m_i)$ and sends it to \mathcal{A} ;

At any time, \mathcal{A} can query the hash values $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ on at most q_h records of its choice $\{\chi_i\}$ and \mathcal{B} responds with random values.
3. *Output.* Eventually, \mathcal{A} outputs a forgery $(\chi_i^*, m_i^*, \sigma_i^*)$ and wins the game if:
 - a. $(\chi_i^*, m_i^*, \sigma_i^*)$ is not any of q_t queries; and
 - b. $(\chi_i^*, m_i^*, \sigma_i^*)$ is a valid tag for $\xi_i^{(2)}$, that is, $e(\sigma_i^*, h) = e(\xi_i^{(2)}, H_1) \cdot e(\prod_{j=1}^s u_j^{m_{i,j}^*}, H_2)$.

An adversary \mathcal{A} is said to (ε, q_t, q_h) -break our scheme if \mathcal{A} makes at most q_t tag queries for dynamic operations and q_h hash queries, as well as success probability of game with at least ε . A scheme is (ε, q_t, q_h) dynamically and existentially unforgeable under an adaptively chosen message attack, if there exists no forgery that is susceptible to (ε, q_t, q_h) -break.

We note that the above-mentioned definition captures a stronger version of existential unforgeability than the standard one on signature schemes, as it requires that the adversary cannot even generate a new tag on a previously signed message. Based on this game, we prove that our scheme is (ε, q_t, q_h) dynamically and existentially unforgeable for the Computational Diffie-Hellman (CDH) problem (see Appendix C, available in the online supplemental material), as follows:

Theorem 3. Given the (ε', q') -CDH in a cyclic group $\mathbb{G} \in \mathbb{S}$ with an order p , our audit scheme is an (ε, q_t, q_h) -dynamically existentially unforgeable to resist the attacks of tag forgery for dynamic data operations with random oracle model, whenever $q_t + q_h \leq q'$ and all ε satisfy $\varepsilon \geq \varepsilon' / (1 - \frac{q_t}{p})$ if each χ_i is collision resistant in IHT χ .

7 PERFORMANCE AND EVALUATION

It is obvious that enormous audit activities would increase the computation and communication overheads of our audit service. However, the less frequent activities may not detect anomalies in a timely manner. Hence, the scheduling of audit activities is significant for improving the quality of audit services. To detect anomalies in a low-overhead and timely manner, we attempt to optimize the audit performance from two aspects: Performance evaluation of probabilistic queries and scheduling of periodic verification. Our basic idea is to maintain a tradeoff between overhead and accuracy, which helps us improve the performance of audit systems.

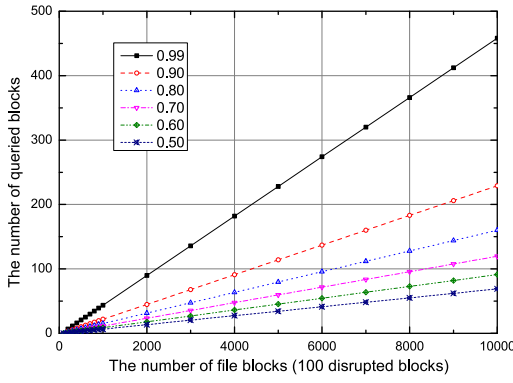


Fig. 9. Number of queried blocks under different detection probabilities and different numbers of file blocks.

7.1 Probabilistic Queries Evaluation

The audit service achieves the detection of CSP servers' misbehaviors in a random sampling mode 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 a timely manner. Assume TPA modifies e blocks out of the n -block file. The probability of disrupted blocks is $\rho_b = \frac{e}{n}$. Let t be the number of queried blocks for a challenge in the protocol proof. We have detection probability $P = 1 - (\frac{n-e}{n})^t = 1 - (1 - \rho_b)^t$. Hence, the number of queried blocks is $t = \frac{\log(1-P)}{\log(1-\rho_b)} \approx \frac{P \cdot n}{e}$ for a sufficiently large n .⁵ 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 . In Fig. 9, we show the results of the number of queried blocks under different detection probabilities (from 0.5 to 0.99), different number of file blocks n (from 10 to 10,000), and constant number of disrupted blocks (10 for $n < 1,000$ and 100 for $n \geq 1,000$).

We observe the ratio of queried blocks in the total file blocks $w = \frac{t}{n}$ under different detection probabilities. Based on our analysis, it is easy to determine that this ratio holds because $w = \frac{t}{n} = \frac{\log(1-P)}{n \cdot \log(1-\rho_b)} \approx \frac{P}{e}$. However, the estimation of w is not an accurate measurement. To clearly represent this ratio, Fig. 10 plots w 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. 10a 100 disrupted blocks, the TPA asks for $w = 4.5$ and 2.3 percent of n ($n > 1,000$) to achieve P of at least 99 and 90 percent, respectively. However, this ratio w is also inversely proportional to the number of disrupted blocks e . For example, in Fig. 10b if there exist 10 disrupted blocks, the TPA needs to ask for $w = 45\%$ and 23 percent of n ($n > 1,000$) to achieve the same P , respectively. It demonstrates our audit scheme is effective under the higher probability of disrupted blocks.

Note that, instead of probability verification, our IPOR scheme also supports an absolute integrity verification, in which TPA checks all file blocks in a challenge query $Q = \{(i, v_i)\}_{i \in I}$, that is, $I = [1, n]$. Furthermore, to shorten

the length of challenge query Q , we simply send a random *seed* to CSP, and then CSP generates the challenge index coefficient pair $(i, v_i) = (i, H_{seed}(i))$ for all $i = [1, n]$, where $H_k(\cdot)$ is a collision-resistant hash function.

7.2 Schedule of Periodic Verification

Too frequent audits may waste the network bandwidth and computing resources of TPA and CSPs. However, less frequent audits would not be conducive to detect the exceptions in a timely manner. Thus, it is necessary to disperse the audit tasks (in which the total number of queried blocks is evaluated, as we discussed in the previous section) throughout the entire audit cycle so as to balance the overload and increase the difficulty of attacks in a relatively short period of time.

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

Also, we make use of the audit frequency f to denote the number of occurrences of an audit event. This means that the number of TPA's queries is $T \cdot f$ in an audit period T . Our evaluation indicates the detection probability $P = 1 - (1 - \rho_b)^{n \cdot w}$ in each audit event. Let P_T denotes the detection probability in an audit period T . Hence, we have the equation $P_T = 1 - (1 - P)^{T \cdot f}$. Given $1 - P = (1 - \rho_b)^{n \cdot w}$, the detection probability P_T can be denoted as $P_T = 1 - (1 - \rho_b)^{n \cdot w \cdot T \cdot f}$. Based on this equation, TPA can obtain the probability ρ_b depending on the transcendental knowledge of the cloud storage provider. Moreover, the audit period T can be predefined by a DO in advance. Hence, the above equation can be used to analyze the parameter values 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. 11, we show the relationship between f and w under 10 disrupted blocks for 10,000 file blocks. We can observe a marked drop of w along with the increase of frequency.

In fact, the relationship between f and w is comparatively stable for P_T , ρ_b , and n due to $f \cdot w = \frac{\log(1-P_T)}{n \cdot T \cdot \log(1-\rho_b)}$. TPA should choose an appropriate frequency to balance the overhead. For example, if $e = 10$ blocks in 10,000 blocks ($\rho_b = 0.1\%$), then TPA asks for 658 blocks and 460 blocks for $f = 7$ and 10 to achieve at least 99 percent of P_T . Hence, an appropriate audit frequency would greatly reduce the sampling numbers, as well as computation and communication overheads of an audit service.

7.3 Implementation and Experimental Results

To validate our approaches, we have implemented a prototype public audit service. Our prototype utilizes three existing services/applications: Amazon Simple Storage

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

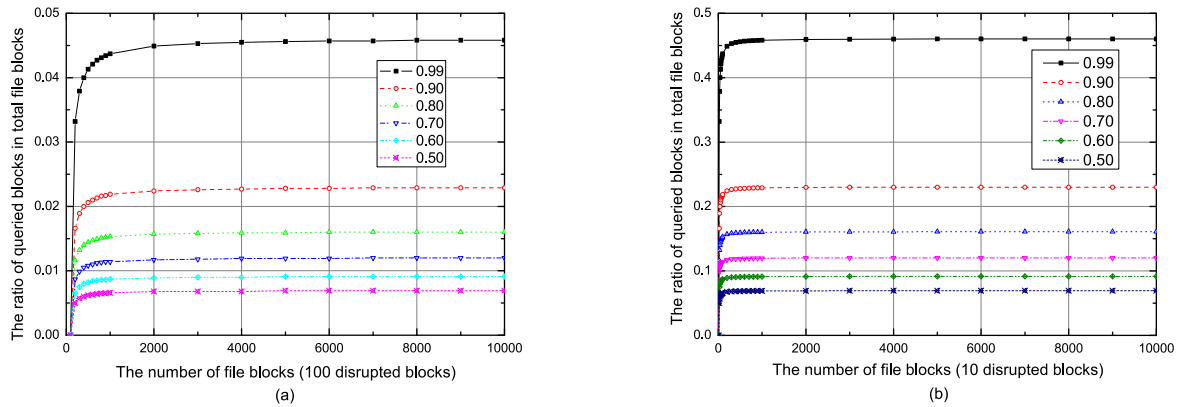


Fig. 10. Ratio of queried blocks under different detection probabilities and different number of disrupted blocks.

Service (S3), which is an untrusted data storage server; a local application server, which provides our audit service; and an existing open source project called pairing-based cryptography (PBC) library upon which to build our prototype. We present some details about these three components as follows:

- *Storage service.* Amazon Simple Storage Service (S3) is a scalable, pay-per-use online storage service. Clients can store a virtually unlimited amount of data, paying for only the storage space and bandwidth that they are using, without an initial start-up fee. The basic data unit in S3 is an object, and the basic container for objects in S3 is called a bucket. In our example, objects contain both data and metadata (tags). A single object has a size limit of 5 GB, but there is no limit on the number of objects per bucket. Moreover, a script on Amazon Elastic Compute Cloud (EC2) is used to provide the support for verification protocol and dynamic data operations.
- *Audit service.* We used a local IBM server with two Intel Core 2 processors at 2.16 GHz running Windows Server 2003. Our scheme was deployed in this server, and then the server performs the integrity check in S3 storage, conforming the assigned schedule via 250 MB/sec of network bandwidth. A socket port was also opened to support the applications' accesses and queries for the audit service.

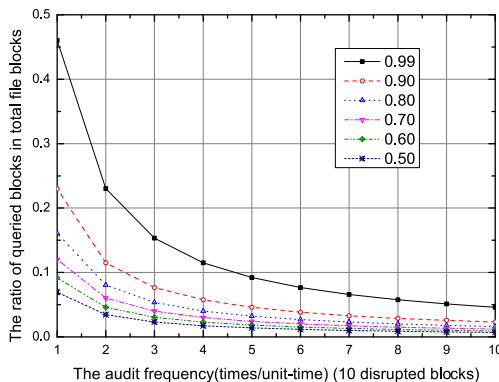


Fig. 11. Ratio of queried blocks in total file blocks under different audit frequency for 10 disrupted blocks and 10,000 file blocks.

- *Prototype software.* Using GMP and PBC libraries, we have implemented a cryptographic library upon which temporal attribute systems can be constructed. These C libraries contain approximately 5,200 lines of codes and were tested on both Windows and Linux platforms. The elliptic curve utilized in our experiments is a MNT curve, with a base field size 159 bits and the embedding degree 6. The security level is chosen to be 80 bits, which means $|p| = 160$.

First, we quantified the performance of our audit scheme under different parameters, such as file size sz , sampling ratio w , and sector number per block s . Our analysis shows that the value of s should grow with the increase of sz to reduce computation and communication costs. Thus, 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 size, and the sampling ratios were also changed from 10 to 50 percent. The experimental results are shown in Fig. 12a. These results indicate that computation and communication costs grow slowly with increase of file size and sampling ratio.

Next, we compared the performance of each activity in our verification protocol. It is easy to derive theoretically that the overheads of "commitment" and "challenge" resemble one another, and the overheads of "response" and "verification" also resemble one another. To validate such theoretical results, we changed the sampling ratio w from 10 to 50 percent for a 10-MB file and 250 sectors per block. In Fig. 12b, 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.

Then, in the Amazon S3 service, we set that the size of block is 4K bytes and the value of s is 200. Our experiments also show that, in TagGen phase, the time overhead is directly proportional to the number of blocks. Ideally, this process is only executed when a file is uploaded into the S3 service. The verification protocol can be run in approximately constant time. Similarly, three dynamic data operations can be performed in approximately constant time for any block.

Finally, reducing the communication overheads and average workloads is extremely critical for an efficient audit schedule. With probabilistic algorithm, our scheme is able to realize the uniform distribution of verified sampling

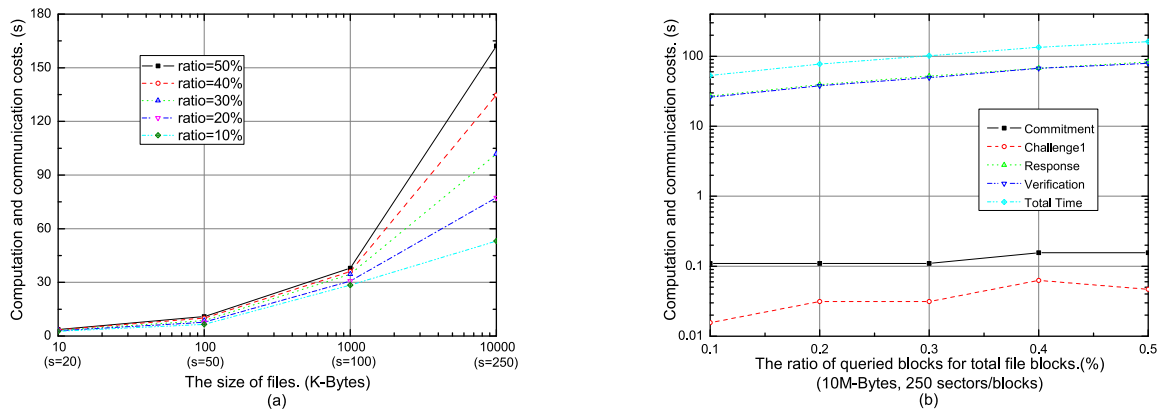


Fig. 12. Experiment results under different file size, sampling ratio, and sector number.

blocks based on the security requirements of clients, as well as the dependabilities of storage services and running environments. In our experiments, we make use of a simple schedule to periodically manage all audit tasks. The results show that audit services based on our scheme can support a great deal of audit tasks, and the performance of scheduled audits are more preferable than the straightforward individual audit.

8 CONCLUSIONS

In this paper, we presented a construction of dynamic audit services for untrusted and outsourced storages. We also presented an efficient method for periodic sampling audit to enhance the performance of TPAs and storage service providers. Our experiments showed that our solution has a small, constant amount of overhead, which minimizes computation and communication costs.

ACKNOWLEDGMENTS

The work of Yan Zhu and Chang J. Hu was supported by the National Natural Science Foundation of China (Project Nos. 61170264 and 10990011) and the National 973 Project (No. 2013CB329606). The work of Gail-J. Ahn and Hongxin Hu was partially supported by grants from the US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308). The work of Stephen S. Yau and Ho G. An was partially supported by a grant from the US National Science Foundation (NSF-CCF-0725340). An earlier version of this paper, entitled "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," appeared in the *Proceedings of the 26th Annual ACM Symposium on Applied Computing (SAC)*, pp. 1550-156, 2011. The authors thank Shimin Chen, a master's student at Peking University, for verifying the scheme by C++ Language.

REFERENCES

- [1] Amazon Web Services, "Amazon S3 Availability Event: July 20, 2008," <http://status.aws.amazon.com/s3-20080720.html>, July 2008.
- [2] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," *Proc. ACM Conf. Computer and Communications Security (CCS '07)*, pp. 584-597, 2007.
- [3] M. Mowbray, "The Fog over the Grimpen Mire: Cloud Computing and the Law," Technical Report HPL-2009-99, HP Lab., 2009.
- [4] A.A. Yavuz and P. Ning, "BAF: An Efficient Publicly Verifiable Secure Audit Logging Scheme for Distributed Systems," *Proc. Ann. Computer Security Applications Conf. (ACSAC)*, pp. 219-228, 2009.
- [5] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, and D.X. Song, "Provable Data Possession at Untrusted Stores," *Proc. 14th ACM Conf. Computer and Comm. Security*, pp. 598-609, 2007.
- [6] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," *Proc. Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm)*, pp. 1-10, 2008.
- [7] C.C. Erway, A. K p cu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," *Proc. 16th ACM Conf. Computer and Comm. Security*, pp. 213-222, 2009.
- [8] H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology Advances in Cryptology (ASIACRYPT '08)*, J. Pieprzyk, ed., pp. 90-107, 2008.
- [9] H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun, and B.-Y. Yang, "A Study of User-Friendly Hash Comparison Schemes," *Proc. Ann. Computer Security Applications Conf. (ACSAC)*, pp. 105-114, 2009.
- [10] A.R. Yumerefendi and J.S. Chase, "Strong Accountability for Network Storage," *Proc. Sixth USENIX Conf. File and Storage Technologies (FAST)*, pp. 77-92, 2007.
- [11] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S.S. Yau, "Efficient Provable Data Possession for Hybrid Clouds," *Proc. 17th ACM Conf. Computer and Comm. Security*, pp. 756-758, 2010.
- [12] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity Auditing of Outsourced Data," *Proc. 33rd Int'l Conf. Very Large Databases (VLDB)*, pp. 782-793, 2007.
- [13] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," *Proc. IEEE INFOCOM*, pp. 1-9, 2010.
- [14] B. Sotomayor, R.S. Montero, I.M. Llorente, and I.T. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14-22, Sept./Oct. 2009.
- [15] A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley, "Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware," technical report, <http://lucene.apache.org/hadoop/>, 2005.
- [16] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," *Proc. Advances in Cryptology (CRYPTO '04)*, pp. 41-55, 2004.
- [17] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Advances in Cryptology (CRYPTO '01)*, pp. 213-229, 2001.
- [18] C.-P. Schnorr, "Efficient Signature Generation by Smart Cards," *J. Cryptology*, vol. 4, no. 3, pp. 161-174, 1991.
- [19] S. Goldwasser, S. Micali, and R.L. Rivest, "A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks," *SIAM J. Computing*, vol. 17, pp. 281-308, Apr. 1988.
- [20] D. Boneh and X. Boyen, "Short Signatures without Random Oracles and the SDH Assumption in Bilinear Groups," *J. Cryptology*, vol. 21, pp. 149-177, Feb. 2008.



Yan Zhu received the PhD degree in computer science from Harbin Engineering University, China, in 2005. He is currently a professor in the School of Computer and Communication Engineering at the University of Science and Technology Beijing, China. From 2007 to 2012, he was an associate professor of computer science at the Institute of Computer Science and Technology, Peking University, China. From 2008 to 2009, he was a visiting

associate professor in the Department of Computer Science and Engineering, Arizona State University, Tempe. He was a visiting research investigator in the Department of Computer and Information Science, University of Michigan, Dearborn, in 2012. His research interests include cryptography and network security. He is a member of the IEEE Computer Society.



Gail-Joon Ahn received the PhD degree in information technology from George Mason University, Fairfax, Virginia, 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 US National Science Foundation, the National Security Agency, the US Department of Defense, the US Department of Energy, Bank of America, Hewlett Packard, Microsoft, and the Robert Wood Johnson Foundation. He received the US Department of Energy CAREER Award and the Educator of the Year Award from the Federal Information Systems Security Educators Association. He is a senior member of the IEEE.



Hongxin Hu received the PhD degree in computer science from Arizona State University, Tempe, in 2012. He is an assistant professor in the Department of Computer and Information Sciences at Delaware State University. His current research interests include access control models and mechanisms, security and privacy in social networks, security in cloud and mobile computing, network and system security, and secure software engineering. He

is a member of the IEEE.



Stephen S. Yau received the BS degree from National Taiwan University, and the MS and PhD degrees from the University of Illinois, Urbana, all in electrical engineering. He is a professor of computer science and engineering and the director of the 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. He is a fellow of the IEEE.



Ho G. An received the BS degree from Iowa State University, Ames, in 2006 and the MS and PhD degrees from Arizona State University in 2008 and 2012, respectively, all in computer science. He has been a software engineer at Google, Inc. since 2012. His research interests include service and cloud computing, security, and privacy.



Chang Jun Hu received the PhD degree from Peking University, Beijing, China, in 2001. He is currently a professor at the School of Computer and Communication Engineering, University of Science and Technology, Beijing, China. His main research interests include parallel computing, parallel compilation technology, parallel software engineering, network storage system, data engineering, and software engineering.