Public Auditing for Trusted Cloud Storage Services

Hui Tian and Yuxiang Chen | National Huaqiao University
Hong Jiang | University of Texas at Arlington
Yongfeng Huang | Tsinghua University
Fulin Nan and Yonghong Chen | National Huaqiao University

Cloud storage can provide on-demand outsourcing of data services for organizations and individuals. However, because customers may not fully trust that cloud service providers meet their legal expectations for data security, techniques for auditing the cloud have attracted increasing attention. Here, we present an architecture of public data auditing, review existing methods or mechanisms for various auditing objectives, and discuss trends and possible future developments.

As the newest computing platform, cloud computing is being envisioned as a utility via which scalable data storage and application services can be massively provided with high efficiency and minimal management overhead. Cloud storage is an important and integral part of the cloud computing ecosystem, with the goal of making it possible to outsource large amounts of data such that both individuals and organizations can enjoy highly virtualized infrastructures while avoiding committing large capital outlays.[1,2] Because of the broad benefits of cloud storage, a growing number of data owners (DOs) tend to outsource their data storage to well-known cloud service providers (CSPs), including Amazon S3, Dropbox, Microsoft SkyDrive, and Google Drive. This has enabled a rapid development of cloud storage and its relevant techniques.

Despite their tremendous advantages in performance, cost effectiveness, and reliability, cloud storage infrastructures still face both internal and external security threats. Data security incidents occur from time to time, such as the downtime of Amazon S3 and Gmail's mass email deletions. The situation can become more severe when some dishonest CSPs motivated by profit incentives (e.g., saving storage space and increasing service volumes) neglect to keep or even deliberately delete ordinary users' rarely accessed data.[2] Therefore, to guarantee the integrity of outsourcing data and strengthen DOs' confidence, it is crucial to develop cloud storage auditing (CSA), also referred to as *remote data auditing*, techniques[1,2,4–29] that can effectively and securely verify whether the cloud is honestly and correctly storing DOs' outsourced data. Moreover, CSA is also an important advanced procedure for cloud forensics, because a failed audit would trigger the forensic process.[3]

In the last few years, a lot of fruitful CSA-related studies have been conducted. Sookhak et al.[1] presented a comprehensive survey involving CSA techniques developed

Digital Object Identifier 10.1109/MSEC.2018.2875880
Date of publication: 20 March 2019

 Copublished by the IEEE Computer and Reliability Societies

before 2014. However, CSA is a fast-moving field because of the dynamic and quickly evolving nature of the cloud environment. The years since 2014 have witnessed not only some more sophisticated schemes[8,10,11,14–20,29] for the existing problems but also many novel techniques addressing the emerging challenges (e.g., dynamic multiple-replicas auditing[21,22] and shared-data auditing[23–28]). Thus, this article presents a more comprehensive and up-to-date study of current CSA developments and, we hope, spur a call to action to motivate further research on effective auditing techniques to address evolving concerns and growing requirements for dependable and trustworthy cloud storage.

Problem Statement for CSA

Architecture

There are generally two CSA models: private auditing and public auditing. The former is the initial model for remote checking of data integrity, with the auditing operation performed directly between DOs and CSPs at a relatively low cost. However, it cannot provide convincing auditing results because DOs and CSPs often mistrust each other. Moreover, it is inadvisable for DOs to carry out the audit frequently because the overhead incurred can be too high to afford.

In contrast, the latter introduces an externally authorized third-party auditor (TPA) to perform the verification work, thereby offering dependable auditing results and significantly relieving DOs of the unnecessary burden.[1,2,7,9] Thus, public auditing is believed to be more rational and practical and has garnered significant attention from the research community.[4–29] Following this trend, we mainly explore public-auditing techniques for cloud storage in this article.

Figure 1 depicts the architecture of the public-auditing model for cloud storage, which involves three entities: the CSP, client, and TPA. CSPs aim to manage and coordinate a number of cloud servers to provide scalable and on-demand outsourcing of data services. Clients, including DOs and users, are the CPSs' customers. Both DOs and users can be individuals or organizations. However, they play different roles in cloud-based data-sharing scenarios. The DO initially stores the data in the cloud to alleviate the burden of local data storage and maintenance, while each user can access and modify the shared data after being granted the appropriate access privileges by the DO. Hence, the DO has the sole right to manage the membership of authorized users.

In the cases where the shareable properties of cloud data are unavailable or not being considered, the DO can be seen as the only authorized user, and other external users are nonexistent. Because of the loss of local data possession, DOs always want to ascertain the correctness of their data stored in the cloud. To build trust between DOs and CSPs, a TPA is introduced to credibly verify the reliability of the cloud storage services. In fact, the audit result released by a TPA would not only help DOs evaluate the risk posed by the procured cloud storage services but would also be beneficial for CSPs to improve the security of their cloud storage platforms.

Desirable Proprieties

In public-auditing schemes, we assume that the TPA is credible but curious, an entity that can perform the audit reliably but may be interested in learning the actual content of the DOs' data. Besides, we consider the CSP to be dishonest in that it may choose to hide the fact that some data are corrupted, which would be motivated by self-interest. Specifically, the CSP may launch the following attacks on the TPA:

1) *forge attack*: the CSP may forge the proofs (challenged data blocks and/or their tags) to deceive the TPA
2) *replacing attack*: the CSP may want to pass verification by replacing the proofs of a corrupted data block and its tag with other ones
3) *replay attack*: the CSP may attempt to pass verification using the proofs generated from the previous ones or other prior information.

To achieve secure and efficient public auditing for cloud storage, an ideal public-auditing scheme should not only address these security threats but

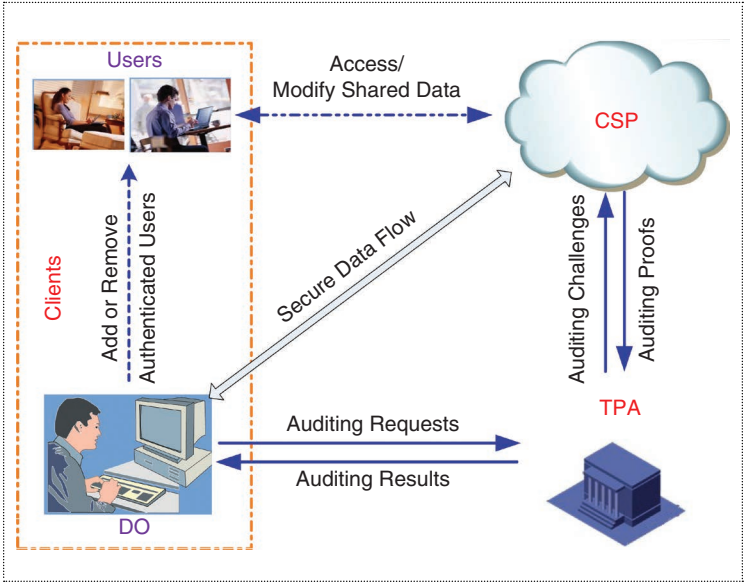

Figure 1. The system architecture of public auditing.

also take the following desirable proprieties into consideration.

- *Support for privacy preservation*: Privacy protection has always been an important requirement for cloud storage. In public auditing, the core of this problem is how to preserve customers' privacy while introducing a TPA. In other words, the TPA should be able to render accurate results without accessing any original data or even learning the actual data content.
- *Support for auditing of dynamic data*: It is well known that a cloud storage system is not just a data warehouse, but one in which customers usually need to update data dynamically due to various application requirements. Thus, public-auditing schemes should support dynamic data operations, such as insertion, deletion, and updating.
- *Support for batch auditing*: To enhance the scalability and efficiency of public-auditing services, the TPA should be able to handle multiple auditing tasks from various clients in a fast and cost-efficient manner.
- *Support for auditing of multiple replicas*: A multiple-replica strategy is commonly adopted in cloud storage to enhance the reliability and availability of data. The TPA should be able to provide strong evidence of multiple replicas being actually and securely stored in the cloud.
- *Support for auditing of shared data*: Because of the popularity of data-sharing services in the cloud, the TPA should also offer effective auditing for shared data. Its major challenges involve the change of users' group membership, identity-privacy preservation, and the traceability of data modification.
- *Lightweight overheads*: The TPA should perform the verification with the minimum communication and computation overheads. Moreover, the auditing process must impose as low overhead as possible on both the cloud server and DOs.

Emerging Approaches for Public Auditing in Cloud Storage

In this section, we explore how to achieve effective public auditing with the desirable proprieties noted previously through reviewing and analyzing the current state of the art. Tables 1 and 2 summarize, respectively, state-of-the-art schemes to meet various auditing requirements and their performance comparison.

Public Auditing with Privacy Preservation

As mentioned previously, public verification is a future trend for CSA that not only can alleviate the overhead of DOs but also provide convincing verification results. To avoid large cloud server costs as well as preserve data privacy, a TPA should conduct effective remote verification without fetching back the data to be verified from the cloud server.[1,2] To achieve this goal, a straightforward solution based on the well-known message authentication codes (MACs) was designed as follows.[5,9]

Prior to data outsourcing, the DO publishes necessary auditing metadata to the TPA, which includes a set of randomly chosen MAC keys and the MACs for each data file. Every time verification for a data file is required, the TPA sends a secret MAC key for the file to the cloud server and requires the cloud server to return a fresh MAC using the given key for checking. In each audit, the communication overhead of this solution (for keys and MACs) is relatively low (only at the bit level). However, there is an obvious deficiency in that the number of prechosen MAC keys limits the total number of times for this low-overhead remote verification. Once all predetermined keys for a data file are exhausted, the DO must retrieve the file from the cloud and publish some new auditing metadata to the TPA, which will inevitably induce an extra burden on the DO.

By contrast, the homomorphic authenticator (HA) technique induces no additional burden on DOs while significantly reducing the communication overhead for transmitting auditing information, making it the most popular approach to achieving public auditing.[4,6–14,16–29] HAs generated from data blocks are unforgeable, and their soundness and correctness have been strictly proved under the well-known computational Diffie-Hellman assumption.[5] Moreover, they can be securely aggregated in such a way that a linear combination of data blocks can be effectively audited by just verifying the corresponding aggregated authenticator.

In public auditing using the HA technique (as illustrated in Figure 2), for each data file, the DO computes HAs for data blocks as the metadata and outsources them to the cloud along with the corresponding file. Once an auditing challenge is received, the CSP computes a linear combination of the challenged data blocks and the aggregation of the corresponding HAs, and it sends them to the TPA as the proof information. Specifically, a data file is divided into n blocks, with each block m_i ($i = 1, 2, \ldots, n$) having a HA σ_i. Whenever it is expected to verify whether the CSP is correctly storing the data, the TPA can launch a challenge $\{(i, v_i) \mid i \in chal\}$ to the CSP by sampling a set of randomly selected data blocks, where *chal* is the index set of the selected blocks. To respond to the challenge, the CSP must return a proof, including a linear combination of all the required data blocks $M = \sum_{i \in chal} v_i m_i$ and an aggregated authenticator $T = \prod_{i \in chal} \sigma_i^{v_i}$. If verification is passed, the cloud data are considered correctly stored with a high probability.

Table 1. The state of the art for various auditing requirements.

State of the Art	Auditing Requirements				
	Privacy Preservation	Dynamic Data	Batch Verification	Multiple Replicas	Shared Data
PDP[4]	×	×	×	×	×
CPOR[5]	×	×	×	×	×
PRC-DPV[6]	√	×	×	×	×
3P-PDP[7]	√	×	—	×	×
3P-LPA[8]	√	√	√	×	×
MHT-PA[9]	×	√	√	×	×
IO-PVA[10]	√	√	√	×	×
RITS-MHT[11]	×	√	—	×	×
IHT-PA[12]	√	√	√	×	×
DAP[13]	√	√	√	×	×
DHT-PA[14]	√	√	√	×	×
DRDA[15]	—	√	—	×	×
EID-PA[16]	—	×	×	×	×
ID-PUIC[17]	×	×	×	×	×
FIB-AR[18]	×	×	×	×	×
IDB-RDIC[19]	√	×	×	×	×
IBDO[20]	×	×	×	×	√
TB-PMDDP[21]	×	√	—	√	×
MuR-DPA[22]	×	√	—	√	×
3P-SPA[23]	√	×	—	×	√
Oruta[24]	√	√	√	×	√
Panda[25]	√	√	√	×	√
PBA-PDP[26]	×	×	√	×	√
SIA-PA[27]	√	√	×	×	√
IBL-PA[28]	√	×	√	×	√
Sed-PA[29]	×	×	—	×	×

Note: *√ means it is supported, × means it is not supported, and — means it was not mentioned.*
PDP: provable data possession; CPOR: compact proofs of retrievability; PRC-DPV: privacy-preserving remote data-integrity-checking protocol with data dynamics and public verifiability; 3P-PDP: privacy-preserving public auditing for secure cloud storage; 3P-LPA: privacy-preserving public auditing protocol for low-performance end devices in cloud; MHT-PA: enabling public auditability and data dynamics for storage security in cloud computing; IO-PVA: efficient public verification of data-integrity for cloud storage systems from indistinguishability obfuscation; RITS-MHT: relative indexed and time-stamped Merkle hash tree-based data auditing protocol for cloud computing; IHT-PA: dynamic audit services for outsourced storage; DAP: an efficient and secure dynamic auditing protocol for data storage; DHT-PA: dynamic hash-table-based public auditing for secure cloud storage; DRDA: dynamic remote data auditing in computational clouds; EID-PA: efficient ID-based public auditing for the outsourced data; ID-PUIC: identity-based proxy-oriented data uploading and remote data-integrity checking; FIB-AR: fuzzy identity-based data integrity auditing; IDB-RDIC: identity-based remote data integrity checking with perfect data privacy preserving; IBDO: identity-based data outsourcing with comprehensive auditing; TB-PMDDP: provable multicopy dynamic data possession in cloud computing systems; MuR-DPA: top-down leveled multireplica Merkle hash tree-based secure public auditing; 3P-SPA: privacy-preserving public auditing for shared cloud data; Oruta: privacy-preserving public auditing for shared data; Panda: public auditing for shared data with efficient user revocation; PBA-PDP: public integrity auditing for dynamic data sharing with multiuser modification; SIA-PA: public integrity auditing for shared dynamic cloud data with group user revocation; IBL-PA: enabling public auditing for shared data in cloud storage supporting identity, privacy, and traceability; Sed-PA: secure auditing and deduplicating data in cloud.

Table 2. A performance comparison of the state of the art.

State of the Art	Overheads						
				Computation			
		Storage		Verification		Updating	
	Communication	CSP	TPA	CSP	TPA	CSP	TPA
PDP[4]	$O(1)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	—	—
CPOR[5]	$O(c+s)$	$O(n)$	$O(1)$	$O(c+s)$	$O(c+s)$	—	—
PRC-DPV[6]	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	—	—
3P-PDP[7]	$O(c)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	—	—
3P-LPA[8]	$O(c)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	$O(t)$	$O(t{\cdot}n)$
MHT-PA[9]	$cO(logn)$	$O(n+2^{logn})$	$O(1)$	$cO(logn)$	$cO(logn)$	$tO(logn)$	$tO(logn)$
IO-PVA[10]	$cO(logn)$	$O(n+2^{logn})$	$O(1)$	$cO(logn)$	$cO(logn)$	$tO(logn)$	$tO(logn)$
RITS-MHT[11]	$cO(logn)$	$O(n+2^{logn})$	$O(1)$	$cO(logn)$	$cO(logn)$	$tO(logn)$	$tO(logn)$
IHT-PA[12]	$O(c+s)$	$O(n)$	$O(n)$	$O(c+s)$	$O(c+s)$	$O(t)$	$O(t{\cdot}n)$
DAP[13]	$O(c)$	$O(n)$	$O(n)$	$O(c)$	$O(c{\cdot}s)$	$O(t)$	$O(t{\cdot}n)$
DHT-PA[14]	$O(c)$	$O(n)$	$O(n)$	$O(c)$	$O(c{\cdot}s)$	$O(t)$	$O(t)$
DRDA[15]	$O(c)$	$O(n)$	$O(n)$	$O(c)$	$O(c)$	$O(t)$	$O(t{\cdot}n)$
EID-PA[16]	$O(c)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	—	—
ID-PUIC[17]	$O(1)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	—	—
FIB-AR[18]	$O(c+s)$	$O(n)$	$O(1)$	$O(c)$	$O(d{\cdot}c+s)$	—	—
IDB-RDIC[19]	$O(c)$	$O(n)$	$O(1)$	$O(c)$	$O(c)$	—	—
IBDO[20]	$O(c+s)$	$O(n)$	$O(1)$	$O(c)$	$O(c+s)$	—	—
TB-PMDDP[21]	$O(c+w{\cdot}s)$	$O(n)$	$O(n)$	$O(c)$	$O(c+w{\cdot}s)$	$O(w{\cdot}t)$	$O(t{\cdot}n)$
MuR-DPA[22]	$cO(logw{\cdot}n)$	$O(wn+2^{logwn})$	$O(1)$	$cO(logw{\cdot}n)$	$cO(logw{\cdot}n)$	$tO(logw{\cdot}n)$	$tO(logw{\cdot}n)$
3P-SPA[23]	$O(c+s)$	$O(n)$	$O(1)$	$O(c)$	$O(c+s)$	—	—
Oruta[24]	$O(c+s)$	$O(d{\cdot}n)$	$O(n)$	$O(c)$	$O(c+s)$	$O(t)$	$O(t{\cdot}n)$
Panda[25]	$O(c+d)$	$O(n+d^2)$	$O(1)$	$O(c)$	$O(d{\cdot}c)$	$O(t)$	$O(t{\cdot}n)$
PBA-PDP[26]	$O(c+m^*)$	$O(n)$	$O(d)$	$O(c{\cdot}s)$	$O(c)$	—	—
SIA-PA[27]	$O(n)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(t)$	—
IBL-PA[28]	$O(c+m^*)$	$O(n)$	$O(d)$	$O(c)$	$O(c)$	—	—
Sed-PA[29]	$O(c)$	$O(s)$	$O(1)$	$O(c)$	$O(c)$	—	—

Note: *n is the whole number of blocks in a file; each block is divided into s segments; c is the number of verified blocks when auditing a file; t is the number of updated blocks; w is the number of copies; d is the number of all authorized users; e is the number of verifications; m is the number of all files; m* is the number of blocks modified; and the — means it was not mentioned.*

It is important to note that, given the huge amounts of data outsourced to the cloud, it is inadvisable to challenge all data blocks for checking the integrity. Instead, it is more affordable and practical for both the TPA and CSP to achieve high-accuracy verification by checking only a portion of the data file, known as *sampling verification*. Generally, if t fractions of the given data are corrupted, the detection probability of the verification by checking randomly sampled c blocks is $P = 1 - (1-t)^c$.[4] In particular, when $t = 0.01$, the TPA only needs to verify 460 randomly chosen blocks to discover this corruption with a probability greater than 99%.

To achieve HA-based auditing, signature techniques based on public key infrastructure (PKI), such as Rivest–Shamir–Adleman[4,6] and Boneh–Lynn–Shacham,[7–14,21–23,25,28,29] are most popularly employed.

Nevertheless, considering the time-consuming certificate management and shortcomings in the security procedures of various certificate authorities,[18,19] some researchers have suggested using an identity-based aggregate signature technique[16–20] for convenient key management, in which the user's public key is generated from his or her identity, and the corresponding secret key can be simply produced by a trusted key generation center. However, identity-based schemes sacrifice some auditing performance compared with those using PKI-based signature techniques. Therefore, it is advisable to choose an appropriate signature technique according to the practical requirements.

Although the actual data are not accessed by the TPA in the described auditing process, the risk of data leakage still exists because, if the TPA has collected enough linear combinations of these blocks, it can easily obtain the sampled data content by solving a system of linear equations.[6,7] To address this concern, an advisable approach is to incorporate random masking—which has two implementation strategies—into the proof generation.[7,8,13,14] In the first strategy, the TPA computes a mask number R with a random number r and a shared global parameter y as $R = y^r$, and it sends R to the CSP together with the challenge. While responding to the challenge, the CSP computes the masked data proof of M as $M' = e(u,R)^M$, where e is a bilinear map and u is a shared global parameter.[13,14] In the other strategy, the CSP chooses a random number r and calculates a mask number $R = yr$, where y is a shared global parameter. Further, the CSP blinds the data proof of M by computing $M' = M + rH(R)$, where H is a hash function. After employing these masking processes, the TPA no longer has enough information to build up a useful system of linear equations for analyzing the actual data content.[7,8] Moreover, because of the algebraic property of the HA, the verification using the proof M' and T can still be effectively performed.

Another approach is to employ a zero-knowledge proof,[19] by which the CSP can convince the TPA to check the integrity of remote files with the knowledge of neither the data blocks nor their corresponding tags. In other words, no information regarding the data content in the response message is returned to the TPA, which thereby achieves perfect data privacy preservation.[19]

Dynamic Auditing

Public auditing should not disregard the dynamic nature of cloud data, in that the latter may be accessed as well as updated frequently for various application purposes. The audit scheme described previously can effectively verify the integrity of all static data. However, it cannot be directly extended to supporting the auditing of dynamic data because it cannot verify the freshness

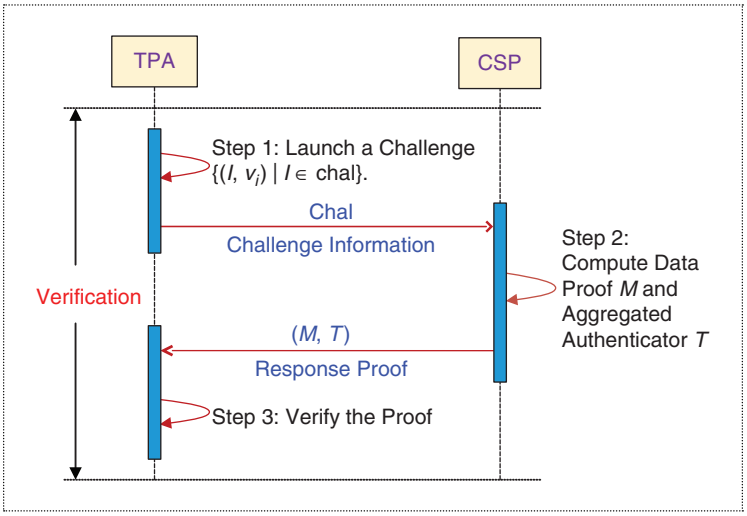

Figure 2. Public auditing using the HA technique.

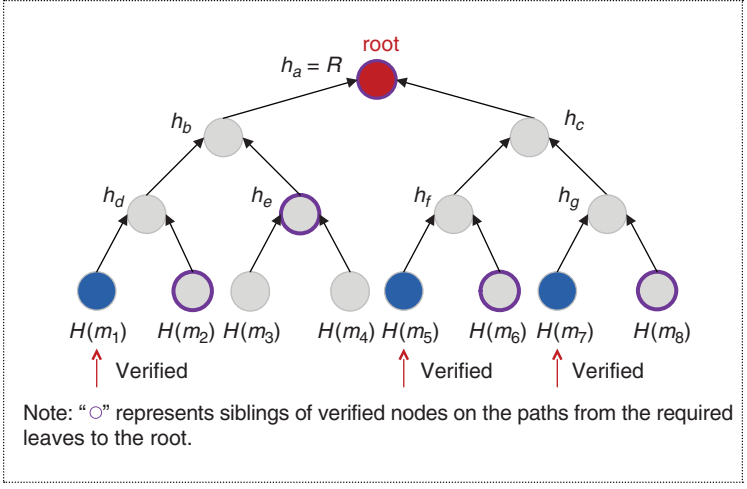

Figure 3. An MHT.

of the cloud data, i.e., whether the CSP is honestly and correctly storing the latest version. Thus, it is crucial to find a new way to achieve dynamic auditing.

To address this concern, authenticated data structures are widely introduced into auditing schemes.[9–15] A Merkle hash tree (MHT), as shown in Figure 3,[9–11] is a typical data structure employed to achieve public auditing for dynamic data. In an MHT-based auditing scheme, the hash values of all data blocks are considered as the leaf nodes of the MHT, and the root R generated from all leaf nodes is a public authentication proof. In the audit phase, the TPA will first compute a root value R' based on the verified nodes and their siblings on the paths from the required leaves to the root received from the CSP, and it will compare R' with R to check the freshness. Assuming there are n leaf nodes in an MHT, the computational complexity for searching a given leaf

Table 3. The index hash table.

Serial Number	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
⋮	⋮	⋮	⋮	
n	n	1	r_n	
n+1	n+1	1	r_{n+1}	← Append

Note: *B_i is the block number, V_i is the version number, and R_i is a random integer.*

node is computed as $O(n)$, which is relatively high.[11] Thus, a modified MHT has been designed recently, with each node containing two values, i.e., the hash value of the data block and the relative index of the node that denotes the number of leaf nodes belonging to the subtree of the given node. In this way, the computational complexity of searching a leaf node can be reduced to $O(\log n)$.[11] However, efficiency and overhead problems still exist in the MHT-based scheme. Specifically, the maintenance of the MHT for data updating would incur extra computational costs to DOs, and the transmission of the involved MHT nodes for verification would also increase the communication overhead.

Another data structure introduced to support data dynamics[12] is the index-hash table (IHT), as shown in Table 3. In an IHT-based scheme, the properties of the data blocks organized in the IHT are stored at the TPA, and their hash values are involved in the generation of the corresponding HAs. Whenever a DO updates a data block, its properties in the IHT and its authenticator must be renewed. If verification is passed, both the integrity and freshness of the cloud data can be considered securely maintained with a high probability because no one can forge all the block properties at the TPA and the authenticators at the CSP. However, this scheme still has some disadvantages: because of the sequential structure of the IHT, updating operations (particularly insertion and deletion) on the IHT are inefficient because they will lead to the adjustment of $N/2$ elements on average, where N is the total number of blocks. Moreover, insertion or deletion operations will inevitably modify the numbers (B_i) of some blocks and lead to the regeneration of their authenticators, which would induce extra computational costs to the DO and unnecessary communication overhead.[13]

To address this problem, we designed a novel 2D data structure called *dynamic hash table* (*DHT*),[14] as shown in Figure 4, to achieve more efficient updating performance. Specifically, to avoid unnecessary adjustments or changes to the other elements while updating, the DHT adopts the following two strategies: first,

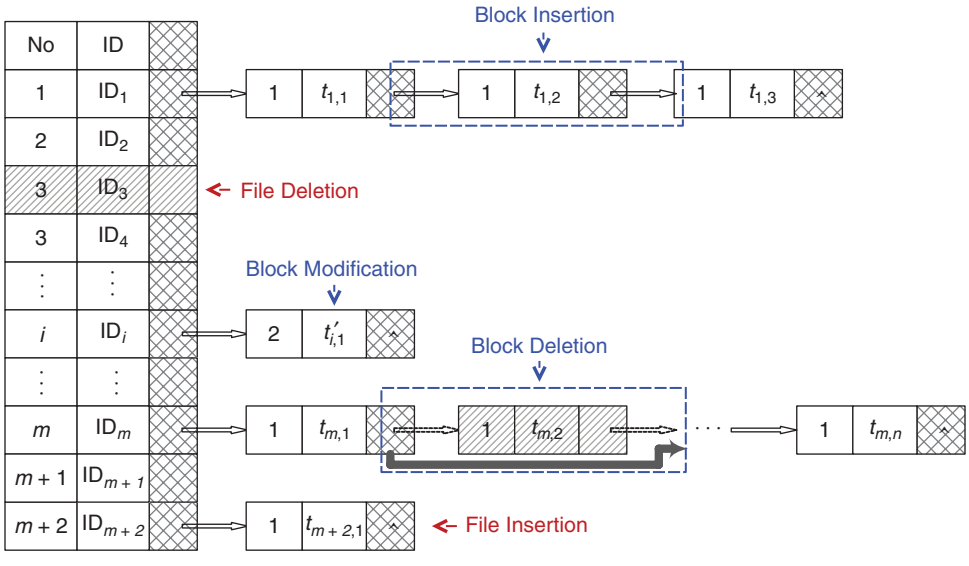

Figure 4. A DHT.

the version information (VI) of each block in the DHT consists of the current version (v_i) of the block and its time stamp (t_i) and does not involve its index number; second, the VI records of the blocks are stored in a link-like structure.

In addition, Sookhak et al.[15] presented a data structure, a divide and conquer table (DCT), as shown in Figure 5, to support dynamic updating of cloud data. The DCT records the logical index and current version for each data block. Like in the IHT, when a block is updated, its record in the DCT and its authenticator should be renewed. Further, to support the dynamic update for large-scale files, the DCT that involves n blocks is divided into k separate data structures with a size of n/k, by which the efficiency for inserting or deleting a block can be significantly enhanced. Specifically, to insert (delete) a new block after the ith block, only $n/k - i$ blocks will be shifted. Another interesting design in the scheme is that the algebraic signature is employed to achieve data auditing; this can reduce the computational overhead on the client and the CSP compared with HA-based schemes, particularly for scenarios with frequent data updating.

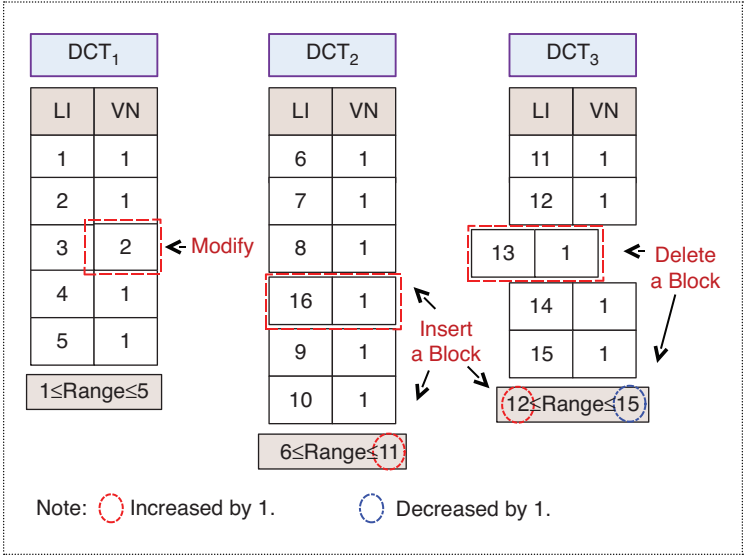

Figure 5. A DCT.

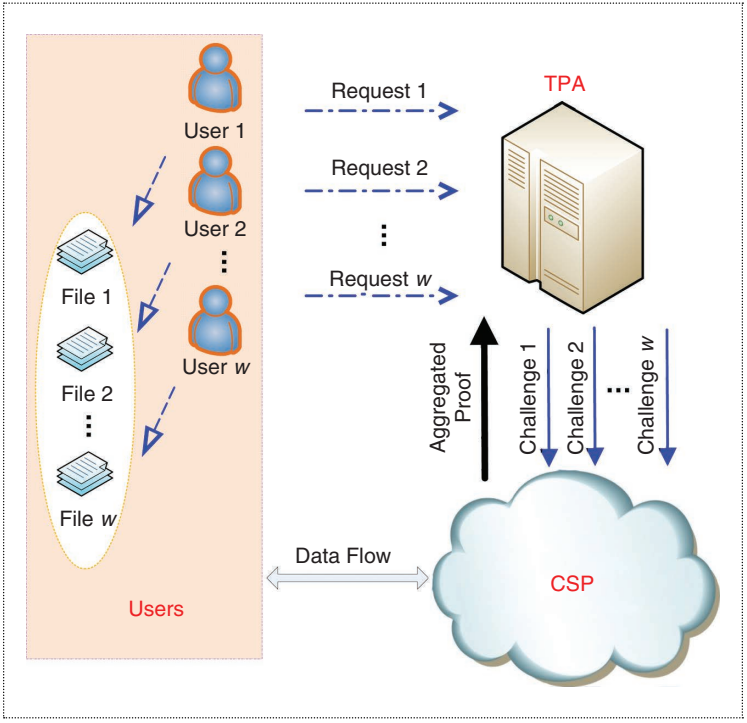

Figure 6. The principle of batch auditing.

Batch Auditing and Multiple-Replica Auditing

In cloud storage, it is common for a TPA to receive auditing requests from many customers simultaneously. It is obviously inefficient for the TPA to perform these tasks one by one. In other words, the TPA should be able to handle all the tasks as a group, i.e., batch auditing.[9,10,12–14,16] To achieve the goal, one can resort to the bilinear aggregate signature technique, which supports the aggregation of multiple signatures by different signers on distinct messages into a single signature and thereby allows synchronous integrality verification for all messages. In this way, as shown in Figure 6, when facing w auditing requests from w different customers, the CSP can send an aggregation of w proofs to the TPA. The TPA can complete all the tasks by verifying the aggregated proof, which can thus significantly save auditing time.

Another problem is how to effectively verify whether the CSP is correctly storing all expected copies of the outsourced data, i.e., multiple-replica auditing, as shown in Figure 7. Because all the replicas are identical, a dishonest CSP may keep only a portion of the copies (or even a single one) but will successfully pass verification using the duplicate proofs. To address this concern, the first intuitive solution would be to use random masking to encrypt the replicas before outsourcing.[21,22]

Specifically, assume that a DO expects a data file F to be stored with w copies in the cloud. Each copy F'_i is divided into n blocks m_{ij} ($i = 1, 2, \ldots, w$; $j = 1, 2, \ldots, n$). The DO first chooses a random function Ψ and generates a random masking $\Psi_{sk}(i\|j)$ for each block m_{ij} using his or her private key (sk). Further, each block m_{ij} is encrypted using $\Psi_{sk}(i\|j)$ so that all replicas are different from one another. Consequently, the probability of the CSP passing verification using duplicate proofs is negligibly small. Moreover, to support the data dynamics, the authenticated data structures described in the "Dynamic Auditing" section have been extended into

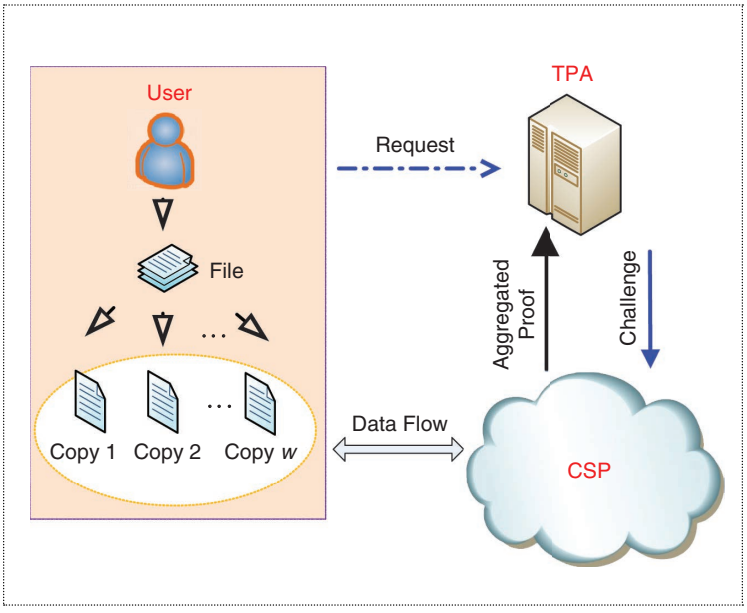

Figure 7. The principle of multiple-replica auditing.

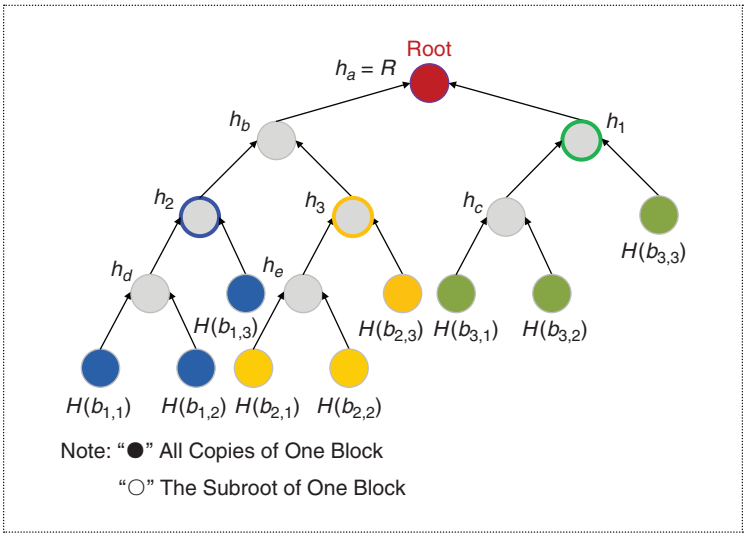

Figure 8. A multiple-replica MHT.

multiple-replica auditing schemes. For example, Barsoum and Hasan[21] introduced the map-version table extended from the IHT to obtain better performance for both the updating and auditing operations. Liu et al.[22] designed an improved MHT structure called *MR-MHT* (as shown in Figure 8), where all copies of each block are organized in a subtree.

Furthermore, because of the potentially large number of replicas, it is advisable for a TPA to audit them in a batch manner, which is similar to batch auditing. In other words, the TPA can verify the aggregated proof received from the CSP only once using the bilinear map technique to effectively reduce both the computational cost of the TPA and communication cost during verification.

Shared-Data Auditing

With the increasing popularity of employing the cloud for teamwork, shared data (which can be accessed and modified by any user of granted group membership) have become an emerging and important branch of cloud data. Besides the requirements mentioned previously, shared-data auditing (as shown in Figure 9) faces the following additional challenges:

1) identity-privacy preservation, i.e., preventing the TPA from learning the information about who last modified which block while auditing[23–25]
2) efficient user revocation, i.e., minimizing the impact induced by the change of the granted group membership[25–27]
3) the traceability of data modification, i.e., allowing the DO to trace who modified which block to achieve the responsibility identification for possible incorrect behaviors.[28]

To preserve identity privacy, Wang et al.[23] first suggested that all authorized users share the same private key for generating authenticators so that only one parameter is involved in the verification. Later, in the auditing scheme called Oruta,[24] the authors used the ring signature to construct authenticators to keep the TPA from knowing the identity of the signer. In both schemes, however, once group membership has been altered, the DO has to update the keys for the changed membership and accordingly regenerate authenticators for all data blocks, which would incur heavy burdens to the DO. To address the concern, the proxy signature is employed to regenerate the authenticators.[23,25,26]

Specifically, assume that sk_{old} is the private key of the revoked user N, and sk_{new} is the private key of another authorized user B designated by the DO. The CSP acting as the proxy is able to regenerate all the block authenticators signed by N using a resigning key $sk_{old \to new}$, which is computed by the DO as $sk_{old \to new} = sk_{new}/sk_{old}$. In these schemes, however, there is a security flaw: the collusion between the revoked user and the CSP would enable the secret keys of other authorized users to be computed. Thus, Jiang et al.[27] presented an auditing scheme based on the group signature with verifier-local revocation to achieve the user revocation as well as prevent collusion between the revoked user and the CSP. Moreover, because of the property of the adopted group signature technique, this scheme can also support identity privacy preservation and the traceability of data modification.

However, the computational overheads of the scheme, especially during the verification process, are much higher than with the other schemes. In addition, to achieve traceability of any data modification, a recent work[28] designed a simple but efficient data structure called an *identity-block list* (*IBL*) for the DO to record the modification information of all data blocks. Unfortunately, this scheme does not take into consideration the regeneration of the authenticator for the blocks modified by the revoked user, making it still vulnerable to security risks; i.e., the CSP or other attackers may forge the authenticators of the involved blocks using the revoked user's keys.

In conclusion, although significant efforts have been made to realize shared-data auditing, no scheme can comprehensively address all challenges while achieving high auditing performance. Thus, auditing of shared data is still an open problem worthy of further study.

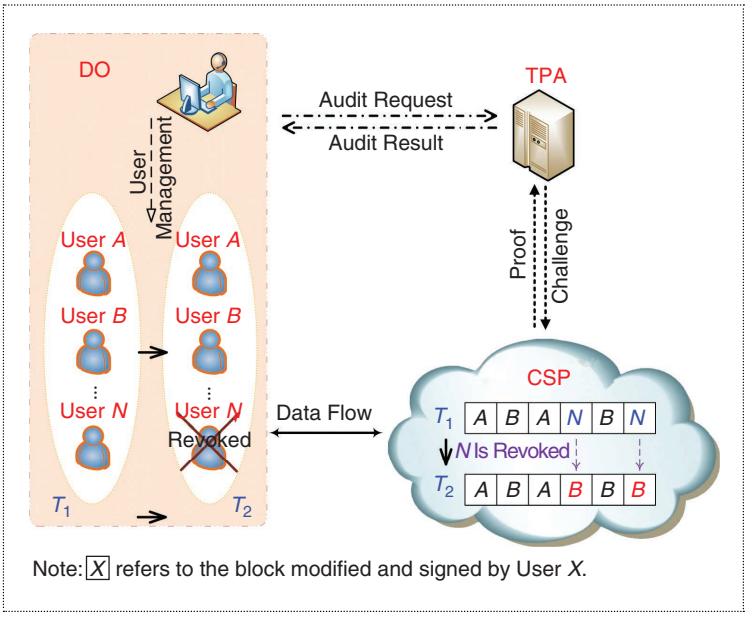

Figure 9. A system model of shared-data auditing.

Challenges and Future Work

We conclude that public auditing will be the right direction for the future development of CSAs. Following this path, state-of-the-art schemes have addressed many significant concerns for CSA. However, from the perspectives of functional completeness and performance enhancement, some unsolved problems and new challenges remain to be addressed in future studies.

Functional-Completeness-Oriented Auditing

Functional completeness is the essential precondition for moving the public auditing from the laboratory into the real world; this still, however, faces two major challenges. The first is the perfection of the auditing techniques for various cloud data. Although elementary auditing for cloud data is now achievable, many fine details remain to be studied or enhanced.

- In batch/multiple-replica auditing, the aggregated verification equation fails with high probability if one or more proofs are erroneous. In this case, how to locate the errors is an important challenge. Compared with one-by-one auditing, binary search is a more effective approach.[21] However, it would still induce heavy computational overheads to the CSP for computing aggregated proofs and to the TPA for performing multiple verifications, not to mention the communication costs for transmitting proof information repeatedly. Therefore, how to locate the errors effectively in batch/multiple-replica auditing remains an open but important issue.
- In multiple-replica auditing, to prevent the CSP from passing verification by keeping a portion of the copies, the DO must generate the required number (w) of different copies for the given files and then transfer them to the CSP, where the overheads for the DO are w times larger than traditional multiple-replica storage. In this sense, the current multiple-replica auditing scheme is not efficient and should be further improved.
- As mentioned in the "Shared-Data Auditing" section, more efforts are still needed to achieve an efficient and functionally complete auditing process for shared data. Moreover, if taking delegatable data outsourcing into account, some specific information, such as outsourcer, file type, and uploading time of the cloud data, should also be auditable, a significant issue for addressing disputes in practice.[20]
- In the cloud storage service, deduplication is popularly used to improve storage utilization as well as reduce the communication overhead of data transfers. Proof of ownership, as a crucial technique for realizing secure deduplication, has attracted increasing attention.[1,29] However, how to achieve effective data auditing while supporting deduplication is still an open problem.

So far, auditing schemes are designed for a specific type of cloud data (e.g., dynamic, multiple-replica, and shared data) and/or particular requirements (e.g., privacy preservation and batching auditing). However, a practical public-auditing scheme should comprehensively support auditing for all types of cloud data and fulfill the corresponding security requirements. Moreover, it should be easily extensible to adapt to constantly emerging new auditing requirements. To achieve these goals, it is critical to construct a unified and scalable

auditing model, which is the other major challenge for achieving functional completeness.

Performance-Oriented Auditing

Performance has always been a significant concern for practical system deployment. Although existing schemes have attempted to improve the performance of both auditing and updating, few studies have demonstrated that the current computational overheads and communication costs for public auditing are affordable for popular personal mobile devices like smartphones and tablets. A recent work[8] first noticed this problem and presented a public-auditing protocol based on online/offline signatures to alleviate computational costs for end mobile devices (also referred to as *edge devices* of the cloud), with limited computing resources when outsourcing data. However, the limited communication capability of edge devices is not taken into account. That is, how to relieve these devices of the stress of communication costs is still an open problem. Moreover, with the continuous emergence of new cloud storage applications, any public-auditing solution will have to be more comprehensive and powerful to cope with the corresponding auditing challenges; however, this inevitably increases the complexity of the verification algorithms. How to achieve a good balance between functional completeness and algorithmic complexity is also an important problem that deserves to be studied further.

Another performance challenge for public auditing stems from the ever-increasing scale of big data in the cloud.[15] A possible solution is designing auditing schemes according to the characteristics of big data. Specifically, because of the large scale and variety of big data, it is advisable to adopt the so-called divide-and-rule strategy, i.e., dividing big data into different categories and designing the most appropriate auditing techniques for each category. For example, for multimedia big data, it may be a good idea to employ the watermarking technique instead of the HA technique to design an auditing scheme; this could not only save the space for storing the tags but also provide relatively high efficiency for data upload or update. For data that are frequently but slightly updated (e.g., micro blogs and consumption records), it is critical to design an auditing scheme that supports fine-grained updating and verification.

Moreover, to satisfy the massive verification requirements for big data, it is necessary to construct a distributed data-auditing model, the advantages of which are twofold. First, an optimal TPA could be selected to conduct the given auditing task with minimal communication costs and the best verification efficiency. Second, if a large number of auditing tasks emerge concurrently, the load balancing among all TPAs can achieve the best overall performance. However, to construct such an ideal auditing model for big data, many concrete problems are worthy of further study.

As a new cutting-edge technology, cloud storage faces many security challenges. One of the biggest concerns is how to determine whether a cloud storage system and its provider meet the legal expectations of customers for data security. To overcome this challenge, CSA techniques are extensively studied and developed. In this article, we first described what CSA is and presented a system architecture for public auditing that we believe to be the right direction for future developments. We then systematically summarized a set of desirable properties for public-auditing services. Further, we provided a comprehensive review of state-of-the-art methods in this field, and identified the pros and cons of these methods. Finally, we explored trends and possible future developments of public-auditing techniques. In view of the function of CSA for bridging the trust gap between customers and CSPs, we believe that it is indispensable for designers of secure cloud storage systems to be cognizant of the various issues and concerns studied in this article. Although it has attracted a great deal of attention, cloud auditing still has a long way to go to realize a practical application, considering the unsolved problems and future challenges. ■

Acknowledgment

This work was supported partly by the National Natural Science Foundation of China under grants U1405254 and U1536115; Program of CSC under grant 201507540001; Program for New Century Excellent Talents in Fujian Province University under grant MJK2016-23; Natural Science Foundation of Fujian Province of China under grant 2018J01093; Program for Outstanding Youth Scientific and Technological Talents in Fujian Province University under grant MJK2015-54; and Promotion Program for Young and Middle-Aged Teacher in Science and Technology Research of Huaqiao University under grant ZQN-PY115.

References

1. M. Sookhak et al., "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," *ACM Comput. Surv.*, vol. 47, no. 4, 2015.
2. C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 220–232, 2012.
3. K. Ruan, J. Carthy, T. Kechadi, and I. Baggili, "Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results," *Digital Invest.*, vol. 11, no. 1, pp. 34–43, 2013.

4. G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Computer and Communications Security (CCS)*, 2007, pp. 598–609.
5. H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances Cryptology (ASIACRYPT '08)*, 2008, pp. 90–107.
6. Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1432–1437, 2011.
7. C. Wang, S. M. Chow, Q. Wang, K. Ren, and W. Lou "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, 2013.
8. J. Li, L. Zhang, J. Liu, K. Ren, and W. Lou, "Privacy-preserving public auditing protocol for low performance end devices in cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2572– 2583, 2016.
9. Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, 2011.
10. Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 676–688, 2017.
11. N. Garg and S. Bawa, "RITS-MHT: Relative indexed and time stamped Merkle hash tree based data auditing protocol for cloud computing," *J. Netw. Comput. Appl.*, vol. 84, pp. 1–13, Apr. 2017.
12. Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic audit services for outsourced storage in clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 227–238, 2013.
13. K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, 2013.
14. H. Tian et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, Dec. 2015.
15. M. Sookhak, A. Akhunzada, A. Gani, M. K. Khan, and N. B. Anuar, "Towards dynamic remote data auditing in computational clouds," *Sci. World J.*, 2014. doi: 10.1155/2014/269357.
16. J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vol. 343, pp. 1–14, May 2016.
17. H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, 2016.
18. Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 72–83, 2019.
19. Y. Yu et al., "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, 2017.
20. Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 940–952, 2017.
21. A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 485–497, 2015.
22. C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: Top-down leveled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2609–2622, 2015.
23. B. Wang, H. Li, and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," in *Proc. IEEE Int. Conf. Communications*, 2013, pp. 539–543.
24. B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, 2014.
25. B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, 2015.
26. J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1717–1726, 2015.
27. T. Jiang, X. Chen, and J. Ma, "Public integrity auditing for shared dynamic cloud data with group user revocation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363– 2373, 2015.
28. G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Software*, vol. 113, pp. 130–139, 2016.
29. J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, 2016.

Hui Tian is a professor in the College of Computer Science and Technology at National Huaqiao University, Xiamen, China. His research interests include network and information security, cloud computing security, and digital forensics. Tian received a Ph.D. in 2010 in computer science from Huazhong University of Science and Technology, Wuhan, China. He is a Member of the IEEE. Contact him at htian@hqu.edu.cn.

Yuxiang Chen is a student at National Huaqiao University, Xiamen, China. Her research interests include cloud

computing security, with a current focus on secure auditing for data outsourcing in public clouds. Chen is pursuing an M.Sc. in computer science at National Huaqiao University. Contact her at yxchen@hqu.edu.cn.

Hong Jiang is chair and Wendell H. Nedderman endowed professor in the Computer Science and Engineering Department at the University of Texas at Arlington. His research interests include cloud computing, big data computing, cloud storage and computer storage systems, parallel input–output, computer architecture and performance evaluation. Jiang received a Ph.D. in computer science in 1991 from Texas A&M University, College Station. He is a Fellow of the IEEE. Contact him at hong.jiang@uta.edu.

Yongfeng Huang is a professor in the Department of Electronic Engineering at Tsinghua University, Beijing. His research interests include cloud computing, cloud storage, network security, and next-generation Internet. Huang received a Ph.D. in computer science in 2000 from Huazhong University of Science and Technology, Wuhan, China. He is a Senior Member of the IEEE. Contact him at yfhuang@tsinghua.edu.cn.

Fulin Nan is a student at National Huaqiao University, Xiamen, China. His research interests include cloud computing security, with a focus on secure auditing for data outsourcing in public clouds. He is pursuing an M.Sc. in computer science at National Huaqiao University. Contact him at flnan@hqu.edu.cn.

Yonghong Chen is a professor in the College of Computer Science and Technology at National Huaqiao University, Xiamen, China. His research interests include network and multimedia information security, cloud computing security, and applied cryptography. Chen received a Ph.D. in 2005 in systems and control engineering from Chongqing University, China. He is a Member of the IEEE. Contact him at iamcyh@hqu.edu.cn.

Digital Object Identifier 10.1109/MSEC.2019.2900871