An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data

Jian Shen, Jun Shen, Xiaofeng Chen, *Senior Member, IEEE*,
Xinyi Huang, and Willy Susilo, *Senior Member, IEEE*

***Abstract*— With the rapid development of cloud computing, cloud storage has been accepted by an increasing number of organizations and individuals, therein serving as a convenient and on-demand outsourcing application. However, upon losing local control of data, it becomes an urgent need for users to verify whether cloud service providers have stored their data securely. Hence, many researchers have devoted themselves to the design of auditing protocols directed at outsourced data. In this paper, we propose an efficient public auditing protocol with global and sampling blockless verification as well as batch auditing, where data dynamics are substantially more efficiently supported than is the case with the state of the art. Note that, the novel dynamic structure in our protocol consists of a *doubly linked info table* and a *location array*. Moreover, with such a structure, computational and communication overheads can be reduced substantially. Security analysis indicates that our protocol can achieve the desired properties. Moreover, numerical analysis and real-world experimental results demonstrate that the proposed protocol achieves a given efficiency in practice.**

***Index Terms*— Auditing protocol, cloud storage, dynamic databases, batch auditing.**

I. Introduction

CLOUD storage, which is one of various cloud services, serves as a practical tool and has made data outsourcing to the cloud an emerging trend. The rapid development of such a cloud service has various causes such as its on-demand outsourcing function, ubiquitous network access, and location-independent resources [1]–[6]. For instance, data are no longer local with cloud storage, ensuring that data owners (DOs) do not have to worry about software or hardware failures. In addition, overhead resulting from maintenance, financial cost, time, and other resources would be greatly reduced, relieving the burden on DOs and local devices.

However, data outsourced to the cloud are not kept securely and still suffer from a variety of security attacks both internal and external [7]–[12]. On the one hand, malicious network attacks, which are external and familiar to Internet users, threaten cloud data. Hackers might retrieve and steal cloud users' data or even corrupt and delete the data, destroying its confidentiality, integrity, and availability. On the other hand, the outsourced data might suffer from cloud service providers' (CSPs') illegal behaviors. In particular, a CSP could secretly delete some data in its storage cycle without authorization to save space for other clients' data. On top of this, a CSP might attempt to obtain the data outsourced to the cloud. Thus, whether the attacks are internal or external, the confidentiality and integrity of cloud data are in danger. Hence, the design of an efficient auditing protocol directed at cloud data has become a research hotspot concerning cloud storage in cloud computing. With auditing, a DO who has already deleted the local copy of data and lost direct control of the data could remotely verify whether their data are stored correctly in the cloud. To make the process of verification substantially more convenient and energy efficient, a new entity named a third-party auditor (TPA) was introduced in, which accepts the auditing delegation from DOs and then executes it. With such an entity, the burden on the client side can be further decreased.

It is widely accepted that technology is advancing and seeing constant development, as is the case for data. Once data are outsourced to the cloud, it will not remain unchanged during the whole period of the cloud [13]–[16]. In other words, a DO might demand to update their cloud data such as with insert, delete, or modify commands. For instance, the data may be incomplete when their owner uploaded them and may need to be improved and perfected at a later time. Over time, some data may not be able to remain current; the cloud owner would want to update the data to the newest version. Another example is whereby, if some data may never be used in the future, the DO should delete the data from the cloud, which is a typical result when using a payment method for cloud services called pay-as-you-go. In other words, dynamic support for cloud data is of great necessity, and some studies have already been conducted on this topic.

Manuscript received December 27, 2016; revised April 3, 2017; accepted May 8, 2017. Date of publication May 18, 2017; date of current version July 20, 2017. This work was supported in part by the National Science Foundation of China under Grant 61672295, Grant 61572382, and Grant U1405254, in part by the State Key Laboratory of Information Security under Grant 2017-MS-10, in part by the China 111 Project under Grant B16037, in part by the Distinguished Young Scholars Fund of Fujian under Grant 2016J06013, in part by the 2015 Project of six personnel in Jiangsu Province under Grant R2015L06, in part by the CICAEET fund, and in part by the PAPD fund. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Shouhuai Xu. *(Corresponding author: Willy Susilo.)*

J. Shen is with the Jiangsu Engineering Center of Network Monitoring, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: s_shenjian@126.com).

J. Shen is with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: sj310310@qq.com).

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710126, China (e-mail: xfchen@xidian.edu.cn).

X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China (e-mail: xyhuang81@gmail.com).

W. Susilo is with the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2705620

Although public auditing protocols with dynamic support directed at cloud data have been researched, an efficient protocol remains to be developed. Although some existing protocols are too expensive in terms of communication and/or computation, other protocols may suffer from low efficiency in supporting data dynamics. Note that in Tian's paper [17], dynamic-hash-table (DHT) is designed to support data dynamics, which is a single linked sequence table. Though Tian's idea is an efficient auditing scheme based on a two-dimensional data structure, it still has some drawbacks. Firstly, the CSP could suffer from collusion attacks from the DO and the TPA, because time stamps for verification are generated by the DO and the auditor only serves the DO. Secondly, no index switcher is designed in [17], which could not indicate the relationship between the index number and the sequence number of a certain data block. Thirdly, the computational cost of the protocol in [17] is still relatively high. Inspired by [17], we introduce a novel dynamic structure composed of a *doubly linked info table* and a *location array* in our auditing protocol, making it substantially more effective.

A. Our Contribution

In this paper, we design an efficient public auditing protocol with novel dynamic structure for outsourced data in the cloud, which preforms better than the state of the art. Note that global and sampling verification is presented to achieve mutual trust between DOs and CSPs. Meanwhile, data dynamics are efficiently provided by the new dynamic structure. In addition, various auditing properties, such as blockless verification and batch auditing, are supported. The main contributions of this paper are summarized as follows.

1) *Global and Sampling Verification Is Proposed in the Protocol:* In practice, cloud and DOs may distrust each other before or during their cooperation. In our protocol, we support *global and sampling verification* to address this issue. Guarantee of *sampling verification* makes owners believe that the cloud has properly stored their data. Meanwhile, *global verification* gives the cloud confidence against owners who are not always behaving appropriately and eliminate the fear of being wrongly accused.
2) *Efficient Data Dynamics With a Novel Dynamic Structure Are Provided in the Protocol:* To the best of our knowledge, we are the first to design a dynamic structure combining a *doubly linked info table* and a *location array* to efficiently support *data dynamics*. The structure successfully handles the relationship between a given data block and its specific location, making data update and batch auditing significantly more convenient. In addition, in practice, owners are not always online for data updates to save time and energy. Hence, although data freshness is reduced, *lazy updates* are supported to reduce overhead.
3) *Various Auditing Properties Are Supported by the Protocol:* Various important properties are established in the proposed auditing protocol to give it greater practical value. *Public auditing* is supported with a trustworthy TPA equipped with professional knowledge introduced to relieve the burden on owners. *Blockless verification* is realized to protect exact outsourced data from CSPs and auditors when auditing tasks are executed. *Batch auditing* is achieved to save time and energy when several DOs would like to verify multiple data files simultaneously.

Notably, the above contributions substantially improves the performance of the designed protocol in this paper without introducing extra communication overhead. Specifically, our protocol is less computationally expensive both in performing single auditing task and batch auditing tasks compared to [17].

B. Related Works

Cloud storage has been studied in recent years as one of the hot spots in cloud computing [18], [19]. Many branches of cloud storage, such as data auditing, privacy preservation, and dynamic updating, are the subject of intense discussion as well.

For data auditing in a cloud, many protocols have been proposed in the past few years and can be divided into private protocols and public protocols. In the model of private auditing protocols, the participating entities are the DO and the CSP. Only the DO possesses the private key, and the entire auditing process is executed by the owner [20]–[22]. However, these solutions increase the burden on DOs, who are not equipped with sufficient computing resources. Moreover, the fatal flaw is that the auditing results are unconvincing because the DO and the CSP distrust each other, and the DO is the sole source of the verification results. To remove the above doubts, a trustworthy TPA is introduced into the system. In 2007, Ateniese *et al.* [23] first proposed the idea of public auditing, which is widely accepted by researchers. After that, increasingly more auditing protocols were designed based on the mechanism of "challenge-proof-verify". In 2013, Wang *et al.* [24] noted that the proposed scheme with public auditability in [23] might leak data information. As a result, the auditing protocol in [24] was designed to be privacy-preserving through the combination of a homomorphic linear authenticator (HLA) and a random masking technique, which is extended to support multiple users as well. One year later, Worku *et al.* [25] suggested that the privacy-preserving auditing protocol in [24] could not preserve the identity privacy of signers. Therefore, the privacy-preserving public auditing scheme proposed by Worku *et al.* employed a ring signature to verify data integrity without exposing signers' identities; this was found to perform better than the work of [24]. Moreover, auditing protocols were extended to key-exposure-resistant protocols by Yu *et al.* [26], [27]. In 2015, they gave the first solution for the exposure of clients' secret keys in [26], where stacks and binary trees were utilized to facilitate key update. The security proof and asymmetric performance evaluation in that paper showed its efficiency and security. Subsequently, they improved and perfected their auditing protocol with key-exposure resistance in 2016 [27], which performed better in the verifiable outsourcing of key updates.

Dynamic support of cloud data has also attracted researchers' attention in recent years. In 2008, Ateniese *et al.* [28] first proposed a partially dynamic provable data possession (PDP) protocol, which represents an important step toward substantially more practical PDP techniques. Inspired by Ateniese's work, Erway *et al.* [29] extended the auditing protocol to support fully dynamic storage with the employment of a skip list, which still achieves a significant improvement in efficiency and privacy. Later, Wang *et al.* [30] employed a Merkle Hash Tree to support full data dynamics, which has been utilized by many researchers. However, both of the above schemes have large communication costs during the updating and verification processes. In 2013, Zhu *et al.* [31] constructed their audit service based on an index-hash table, therein requiring lower computational costs and communication overhead. Nevertheless, update operations, such as insertion and deletion, are inefficient because such an index table is a sequence table, which takes half of the total table length to locate a certain element on average. To address the above problem, Tian *et al.* [17] designed a public auditing protocol based on DHT, and Jian *et al.* [32] designed a protocol using an index switcher, both of which were proposed in 2016. In [17], data information could be stored in the DHT with no restrictions on serial number or block number because no such element is included in the DHT. However, the CSP could suffer from collusion attacks from the DO and the TPA because time stamps for verification are generated by the DO, and the TPA only serves the DO. In [32], an index switcher indicating the relationship between block indices and tag indices could efficiently avoid tag re-computation caused by block update operations. However, such a switcher is transformed among the system periodically, causing significant extra costs. In addition, the proposed index switcher consists of two tables, rather than a complete structure, and does not show how to switch between the two tables.

In contrast to the above solutions, in this paper, we design a substantially more efficient public auditing protocol by employing a novel dynamic structure consisting of a *doubly linked info table* and a *location array*. Note that, compared to [17], the proposed protocol can achieve mutual trust between DOs and CSPs, where global auditing is used to resist collusion attacks from DOs and auditors. Moreover, the *doubly linked info table* is able to locate a certain block more quickly and the *location array* can maintain the relationship between blocks and their specific locations.

C. Organization

The remainder of this paper is organized as follows. Section 2 introduces some preliminary works to allow the reader to obtain a better understanding of the topic. Section 3 presents the system model and security objectives of the proposed protocol. Section 4 shows our dynamic structure in detail. Section 5 displays our detailed protocol. Section 6 presents the correctness analysis, security analysis and performance analysis. The conclusion is drawn in Section 7.

II. Preliminaries

This section introduces some preliminaries to facilitate the reader's understanding, including the bilinear pairing, computational assumption and BLS-based homomorphic verifiable authenticator.

A. Bilinear Pairing and Computational Assumption

Bilinear pairing is a relatively mature technique and has been widely employed in cryptography since 2001, when an identity-based encryption was designed by Shen *et al.* [33], Boneh and Franklin [34], He *et al.* [35], Shen *et al.* [36], and Chen *et al.* [37]. Certainly, such an efficient technique has been utilized in cloud auditing as well. The detailed definition is as follows.

Definition 1 (Bilinear Pairing): Let G_1, G_2, and G_T be multiplicative cyclic groups of a large prime order p. A map function $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear pairing only when it satisfies the three properties below:

1) *Bilinear:* For $\forall g_1 \in G_1$, $\forall g_2 \in G_2$, and $\forall a, b \in Z_p$, there is $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$.
2) *Non-Degeneracy:* For $\exists g_1 \in G_1$ and $\exists g_2 \in G_2$, there is $e(g_1, g_2) \neq 1$.
3) *Computability:* For $\forall g_1 \in G_1$ and $\forall g_2 \in G_2$, there exists an efficient algorithm to compute $e(g_1, g_2)$.

Then, the Computational Diffie-Hellman (CDH) problem is described as follows, the hardness of which is the basis of the security of our protocol.

Definition 2 (CDH Problem): Given g^a and g^b, where g is a generator of G_1 and $a, b \in_R Z_q^*$, compute g^{ab}.

B. BLS-Based Homomorphic Verifiable Authenticator

The final preliminary is the BLS-based Homomorphic Verifiable Authenticator (BLS-HVA), which has been widely employed by a large number of cloud auditing protocols [38]–[40]. The following is the definition.

Definition 3 (BLS-HVA): Given a data file that contains n blocks, $F = \{m_1, m_2, m_3, \cdots, m_i, \cdots, m_n\}$, let G and G_T be two multiplicative cyclic groups of a large prime order p, and let $e : G \times G \rightarrow G_T$ be a bilinear pairing. Select a random secret key $sk = a \in Z_p$, and the corresponding public key is $pk = \{g, v = g^a\}$. Then, BLS-HVA for each data block m_i is $\sigma_i = (h(m_i))^a$, where $h(\cdot)$ is a hash function. For verification, the auditor will simply check whether $e(\prod_{i \in Q} h(m_i), v) = e(\sigma, g)$ holds, where Q is the set of challenged blocks and σ is the aggregated authenticator of these blocks' BLS-HVA.

With such a technique, blockless verifiability can be realized. In addition, two other properties, homomorphism and non-malleability, are both possessed by BLS-HVA [17].

III. Problem Statement

A. The System Model

As described in Fig. 1, the whole model contains three primary entities: the CSP, the DO and the TPA.

The CSP is an entity that provides cloud clients with multiple cloud services and that seems to have inexhaustible storage and computing resources. In our model, the CSP is

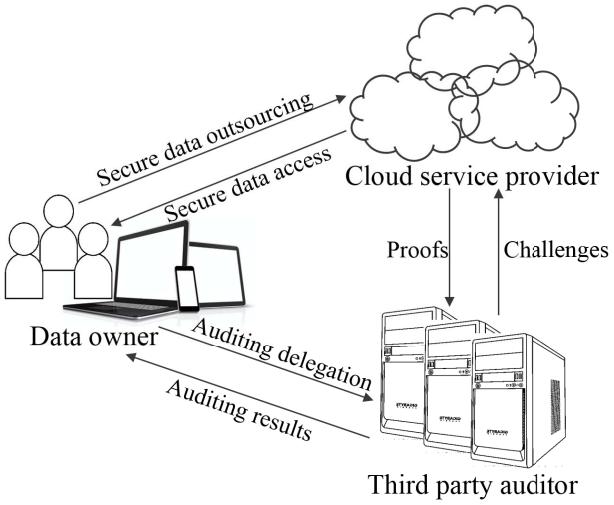

Fig. 1. The system model.

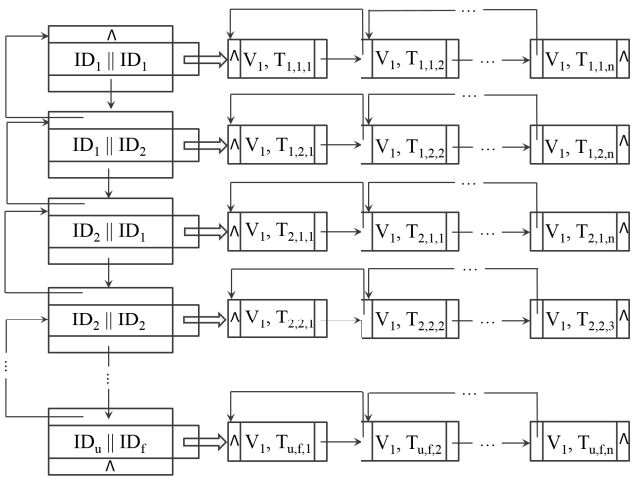

Fig. 2. The doubly linked info table.

responsible for the storage of data from DOs. The DO, which can be an individual person or a major company, is an entity that relies on the resource-rich cloud for data outsourcing. The devices used to communicate with the TPA can be phones, computers, tablets, etc. The final component is the TPA, which is a trustworthy third party between the CSP and the DO. Generally, the TPA is equipped with professional knowledge of data verification so that it can provide convincing auditing results.

As for the relationships among the three entities, a brief exposition will be provided here. The DO outsources its data files to the cloud and obtains them from the cloud as necessary. The data flow between the DO and the CSP should be secure and encrypted because both entities distrust each other. The TPA is picked by the DO and is responsible for auditing on behalf of the DO. When the DO would like to verify a certain data file in the cloud, it would delegate such a task to the TPA. Upon receiving the task, the TPA would send challenges to the CSP, check the response proofs and finally return the auditing results to the DO.

B. Security Objectives

To enable secure and efficient dynamic public auditing for cloud data, our protocol design should achieve the following objectives:

1) *Public auditing:* to allow anyone trusted (not just the DO itself) to verify the cloud data on demand without retrieving a copy of the data.
2) *Storage completeness:* to ensure that the CSP can never pass the auditor's verification when it does not store the owners' data intactly.
3) *Dynamic operations support:* to allow the DOs to perform updates (insertion, deletion and modification) on their outsourced data files while guarantee the efficient public auditing.
4) *Storage freshness:* to allow the cloud and the auditor to keep the latest version of data blocks and corresponding authenticators.
5) *Batch auditing:* to allow the auditor to handle multiple auditing delegations from possibly large number of various DOs simultaneously.

IV. The Proposed Dynamic Structure

In this section, the dynamic structure employed in our protocol, which is a novel structure to the best of our knowledge and is composed of a *doubly linked info table* and a *location array*, will be introduced in detail.

A. Doubly Linked Info Table

The *doubly linked info table* (DLIT) is a two-dimensional data structure employed by the TPA to store data information concerning auditing, differing from the one-dimensional Index Hash Table (IHT). The specific structure of the DLIT is shown in Fig. 2. Data information in the DLIT is divided into two types: file information and block information. The left part is the file information, including user ID and file ID. In previous works by other researchers, such as [17], [24], [31], [41], and [42], the file information simply consists of the file ID, therein making the length of the file ID long. Moreover, when the total number of files increases over time, it becomes more difficult to make each file ID unique. Hence, we use the concatenation of the user ID and file ID to identify each file, making the unique identifier much easier to find. For instance, if the file ID is 4 bits, 16 identifiers can be generated. However, even if a user ID of only one bit is added into the process of identifier generation, the number of identifiers can be doubled to 32. In the real world, both the file ID bit and the user ID bit will be larger, and the number of identifiers will consequently be increased. The right part is the block information, including the current version number and the time stamp, which are generated when a given block is uploaded or updated. Here the version number is denoted as V_{vn}, and the time stamp is represented as $T_{u,f,n}$, where u is the user ID, f is the file ID, and n is the block's index number. In another word, (V_{vn},$T_{u,f,n}$) indicates that for the n-th block of the u-th user's f-th file, the time stamp for the V_{vn}-th version is $T_{u,f,n}$. In the DLIT, both the file information and the

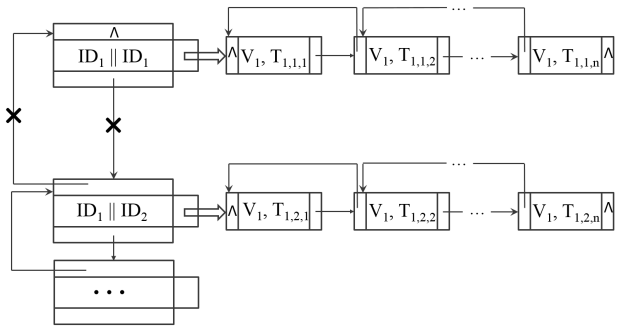

Fig. 3. The status of the DLIT before file insertion.

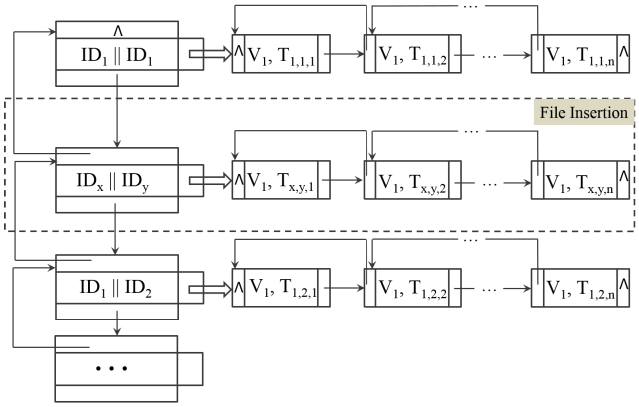

Fig. 4. The status of the DLIT after file insertion.

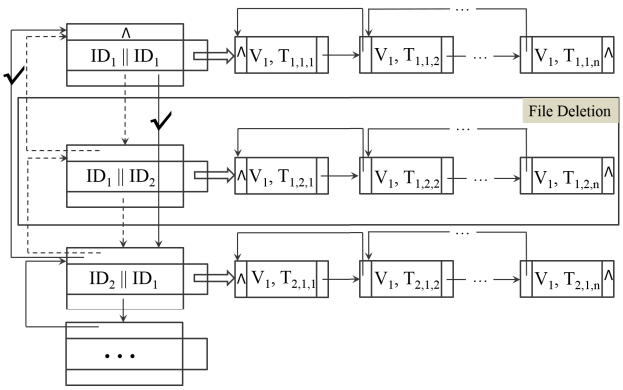

Fig. 5. The change of the DLIT in file deletion.

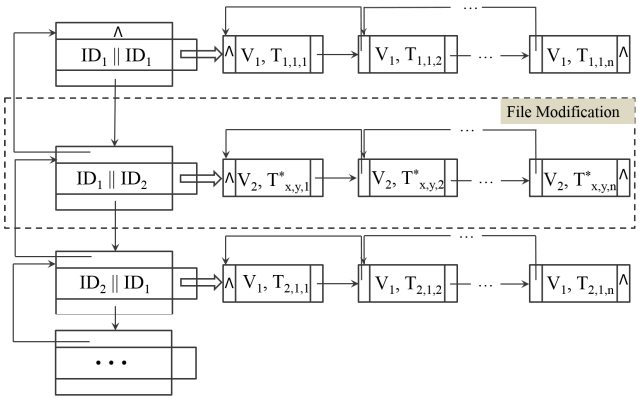

Fig. 6. The change of the DLIT in file modification.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & \cdots \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & \cdots \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & \cdots \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & \cdots \\ \cdots & \cdots & & & & & \end{pmatrix}$$

Fig. 7. The location array.

block information are doubly linked, forward and backward to the next information record and the prior record. With such a double linking data structure, the insertion and deletion of a file or data block will no longer cause a change in other records in the DLIT. Moreover, the advantages of the DLIT will be reflected in batch operations at lower costs when searching for a certain element.

As elements in the DLIT are divided into two types, operations on the information are divided into file operations and block operations; both groups contain insertion, deletion, modification, and search. For simplicity, we only introduce the processes of file operations because block operations are similar to file operations. Fig. 3 and Fig. 4 are statements before and after file insertion, respectively. We assume that user x would like to upload file y onto the cloud. First, the double link between file 1 and file 2 of user 1 should be cut off, which is indicated in Fig. 3 as the two crosses shown. Then, the file to be inserted, together with its blocks, will be doubly linked with the whole table, shown in the File Insertion dashed box of Fig. 4. Fig. 5 describes the process of file deletion, where the file in the File Deletion box will be deleted. As a result, all links with file 2 of user 1 from other files are cut off. Then, new double links of files immediately before and after file 2 of user 1 are generated, as shown as arrows with ticks, upon which file deletion is finished. In Fig. 6, file 2 of user 1 is taken for instance to show a file modification operation. It is clear that the version number and time stamp of the data blocks are updated. The final operation is search. As we all know, to locate a certain file or block, visiting the whole table from the first element is necessary but only for the first round of search. After that, searching elements will start from the current location instead of restarting from the first record again because every element is equipped with a forward pointer and a backward one simultaneously, which outperforms the Dynamic Hash Table in [17].

B. Location Array

The *location array* (LA) is employed as an index switcher in our protocol, shown in Fig. 7, which looks like a common array. In our protocol, the LA is employed to keep the

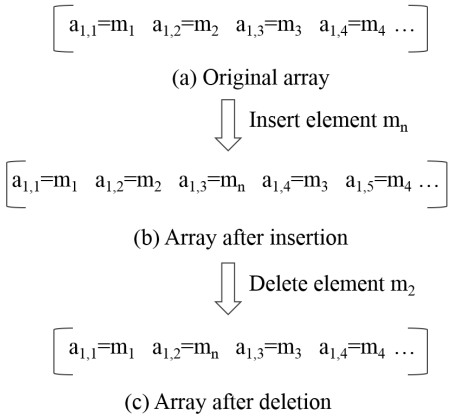

Fig. 8. Operations on the location array.

TABLE I
NOTATION

G, G_T	multiplicative cyclic groups
p	prime order of G, G_T
g,u	generators of G
Z_p	set of nonnegative integers less than p
$h(\cdot) : \{0,1\}^* \rightarrow G$	one-way hash function
$e : G \times G \rightarrow G_T$	a bilinear pairing
F	data file outsourced in cloud
m_i	blocks of a certain data file
Loc_i	specific location of m_i

mapping between index number and serial number (the specific location) of data blocks.

The index of $a_{x,y}$ indicates the specific position of the block with its name stored in $a_{x,y}$: the y-th block of the x-th file. The element m_i in each $a_{x,y}$ is the name of each block, related to its index number. With such an array, the location of a certain data block can be marked with the corresponding index of the array element. Taking $a_{1,4} = m_7$ as a simple example, we can infer that block m_7 is at the location of the 4-th block of the 1-st data file. Thus, the LA can serve as an efficient index switcher.

For the specific operations on the LA, the insertion and deletion of a certain block will affect its content, whereas search and modification will not. Fig. 8 shows changes in an array in regard to element insertion and deletion. Fig. 8(a) is the original array, therein containing blocks of the 1-st file. When a new element m_n is inserted at the 3-rd position, as Figu. 8(b) shows, m_n will be the new $a_{1,3}$, and the original m_3 will be at the 4-th position, etc. Then, element m_2 is deleted from the array in Fig. 8(b), elements after m_2 will move forward one by one, resulting in $a_{1,2} = m_n$, etc.

V. The Proposed Public Auditing Protocol

In this section, we will present the core of our public auditing protocol, which is based on a novel dynamic structure consisting of a *doubly linked info table* and a *location array*. In addition, the detailed introduction of the global verification and sampling verification will be presented in subsection 5.3. After that, the extended batch auditing and the dynamic auditing of our protocol will be described in subsections 5.4 and 5.5, respectively. Note that notations used in our paper are shown in the following.

A. Notations

As is shown in Table I, some primary notations used in our protocol are listed. G and G_T are two multiplicative cyclic groups of a large prime order p, and g and u are generators of G. Z_p is a set of nonnegative integers less than p, and $h(\cdot) : \{0,1\}^* \rightarrow G$ is a one-way hash function from strings to elements in G. The data outsourced to the cloud to be verified are denoted as F, consisting of blocks m_i. For unavoidable insertion and deletion operations during data update, the specific locations of m_i are not always the i-th block of the file. Hence, Loc_i is employed to denote the specific location of m_i.

B. High Description of the Protocol

In this section, the proposed auditing protocol can be divided into two phases: the setup phase and the verify phase. The former phase is responsible for some preparation works and contains three algorithms: KeyGen, Filepro2C, and Filepro2T. The latter contains three algorithms as well: ChalGen, ProofGen, and VerifyProof. The following are the details:

1) Setup Phase: In this phase, some preparations are made for system setup, which are the responsibility of the DO. First, the DO generates the secret and public key pair in KeyGen. Then, some pre-processing tasks for the CSP and TPA are performed by the DO in Filepro2C and Filepro2T, respectively, where "2" means "to". The designs of the three algorithms are shown below.

1) ***KeyGen:*** The DO runs KeyGen to generate public and secret parameters. First, the DO chooses a random signing key pair (ssk, spk) for the signature of the file name. Then, the DO picks $a \in Z_p$ randomly as one part of system secret key and ssk as the other part. Consequently, the secret key would be $sk = (a, ssk)$, which is known only by the DO. Moreover, the DO picks random generators g,$u \in G$, and let $v = g^a$. Then, the system public key would be $pk = (u, g, v, spk)$. In summary, the running result of the algorithm KeyGen is $(sk, pk) = \{(a, ssk), (u, g, v, spk)\}$.
2) ***Filepro2C:*** The DO runs Filepro2C to pre-process files to be outsourced to the cloud. First, the DO divides F into blocks or even sectors, where F represents the cloud file, as mentioned above. For simplicity, we only divide F into n blocks here:

$$F = \{m_1, m_2, \cdots, m_i, \cdots, m_n\},$$

where m_i is the general name of data blocks, and $i \in [1, n]$. Then, the DO generates an authenticator for

each block m_i:

$$\sigma_i = (h(V_i||T_i) \cdot u^{m_i})^a, \tag{1}$$

of which the aggregated set is

$$\sigma = \{\sigma_i\}_{i\in[1,n]}.$$

In the above equation, the mentioned V_i is m_i's version number, and T_i is its time stamp. Moreover, the DO generates file tags based on ssk to ensure the integrity of the unique file information:

$$\vartheta = U_{ID}||F_{ID}||SIG(U_{ID}||F_{ID})_{ssk}. \tag{2}$$

Finally, the DO uploads $\{F, \sigma, \vartheta\}$ to the cloud for storage. At this point, the pre-processing tasks for the CSP have all been completed.

3) ***Filepro2T:*** The DO runs Filepro2T to pre-process information to be stored in the TPA. Specifically, the file information and the block information are required for the *doubly linked info table*, while the specific location of each block is required for the *location array*. Specifically, F_{ID}, U_{ID}, V_i, T_i, and Loc_i are uploaded by the DO to the TPA. Upon receiving all the information, the TPA establishes the DLIT and the LA and then saves them. At this point, the pre-processing tasks for the TPA have all been completed.

2) Verify Phase: The second phase performs data verification, therein involving the DO, CSP, and TPA. Specifically, challenges are launched from the TPA to the CSP in ChalGen, after which the CSP responds in ProofGen to prove the integrity of the cloud data for which it is responsible. In VerifyProof, auditing results are obtained from the TPA's calculation and returned to the DO. The detailed design of each algorithm is as follows.

1) ***ChalGen:*** The TPA runs ChalGen to launch the verification challenge to the CSP on behalf of the DO. First, the DO delegates the verification task of a certain file (or several files, which will be discussed in Section 5.4) to the TPA. Then, the TPA asks the CSP for the corresponding file tag ϑ and verifies the correctness of it by spk. If this fails, the TPA would inform the DO that the file has been corrupted; otherwise, verification continues. Then, the TPA picks s random elements $a_{x,y}$ in the LA and obtains the corresponding blocks' information in the DLIT. Finally, the TPA sends the verification challenge

$$chal = \{i, r_i\}_{i\in[1,s]}$$

to the CSP, where $s \in [1, n]$ and r_i is randomly picked from Z_p.

2) ***ProofGen:*** The CSP runs ProofGen to generate corresponding proofs of the required blocks, which contain two parts: a tag proof indicating the authenticator's correctness and a data proof indicating the data's integrity. More specifically, the CSP generates the tag proof

$$T = \prod_{i\in[1,s]} \sigma_i^{r_i}, \tag{3}$$

and the data proof

$$D = \sum_{i\in[1,s]} m_i \cdot r_i. \tag{4}$$

Upon completion, the complete proof (T, D) will be sent back to the TPA as the response of the verification challenge.

3) ***VerifyProof:*** The TPA runs VerifyProof to check whether the proof returned from the CSP is valid. According to the data information stored in the DLIT, the TPA computes

$$DI_i = e(h(V_i||T_i), v) \tag{5}$$

for each data block to be verified. Then, the aggregated data information will be

$$DI = \prod_{i\in[1,s]} DI_i. \tag{6}$$

Finally, the TPA checks whether

$$e(T, g) = DI \cdot e(u^D, v) \tag{7}$$

holds. If it does, the data outsourced to the cloud is complete; otherwise, the data are not complete.

C. Global and Sampling Verification

In our design, the verify phase includes the global verification and sampling verification. Specifically, the sampling verification is the common one described in the existing auditing schemes, while the global verification is the concept proposed in this paper.

The introduced process above in Section 5.2 is the sampling verification, where the TPA randomly picks several data blocks to establish the challenge. The sampling verification can be conducted irregularly by the auditor to check the completeness and correctness of cloud data, which aims to protect the rights of DOs. Through the sampling verification, resource can be saved under the premise of ensuring the correctness and completeness of cloud data.

In [17], Tian *et al.* employed a time stamp in data auditing in their protocol, which is generated by the DO and maintained by the TPA. In the verification process of the protocol designed in this paper, the time stamp participating in is generated by the DO itself as well. That is where the problem lies. Specifically, the TPA is a trustworthy third party chosen by the DO to undertake the auditing task, which is responsible for the DO to monitor the CSP. In another word, the TPA and the DO are on the same side. There is a strong possibility that the DO will collude with the TPA by uploading a data block mismatching the time stamp. So that, the uploaded data block will never pass the data auditing, and the DO and the TPA can carve up the fraudulent compensation for data corruption from the CSP.

Therefore, we introduce the concept of "global verification" in our protocol to address such a collusion attack mentioned above. Specifically, at the moment that the data file is uploaded or updated in the cloud, all these data should be verified, which is the meaning of "global". If these data uploaded can pass the verification, the CSP will provide

storage services to them; otherwise, the CSP will refuse to store these data. After the global verification, the concerns of the CSP can be reduced.

Certainly, the global verification requires some additional operations in the verify phase. In the ChalGen algorithm, the TPA inserts all blocks' indexes in $chal$

$$chal = \{i, r_i\}_{i \in [1,n]}$$

when the whole file is uploaded or

$$chal = \{i, r_i\}_{i \in [1,c]}$$

when updates appear, where c is the total number of changed blocks. Then, the CSP responds with the corresponding proof in ProofGen, and the follow-up process remains unchanged.

In summary, the sampling verification serves DOs, while the global verification serves the CSP.

D. Batch Auditing

In a real-world scenario, the TPA may not serve only one DO or even be responsible for only one data file. Multiple auditing tasks from various DO delegations may be waiting for processing simultaneously. Hence, batch auditing is introduced into our design to address the above phenomenon.

The batch auditing supported in our work is divided into two categories: one for multiple files from one DO and the other for multiple files from multiple DOs. We suppose that $u_i \in [1, u_s]$ users delegate $f_i \in [1, f_s]$ files' $i \in [1, s]$ blocks' auditing tasks to the TPA simultaneously. In the first situation, $u_i = 1$ so that the verification equation can be

$$\prod_{f_i \in [1,f_s]} e(T, g) \stackrel{?}{=} \prod_{f_i \in [1,f_s]} DI \cdot e(u^D, v). \tag{8}$$

If Eq. (8) holds, all the verified files from a single DO are completely and correctly stored in cloud. Otherwise, at least one data file is corrupted.

In the other type, the TPA would check

$$\prod_{u_i \in [1,u_s]} \prod_{f_i \in [1,f_s]} e(T, g) \stackrel{?}{=} \prod_{u_i \in [1,u_s]} \prod_{f_i \in [1,f_s]} DI \cdot e(u^D, v). \tag{9}$$

If Eq. (9) holds, the completeness and correctness of all the verified files from multiple DOs can be ensured. Otherwise, at least one data file of a DO is corrupted.

E. Dynamic Auditing

To support efficient dynamic operations on data along with its information, we design the DLIT and LA in our protocol. Changes in the two objects have been mentioned briefly in Section 4 when file-layer updates occur. In this section, the specific operations are introduced, involving block insertion (B_{insert}), block deletion (B_{delete}) and block modification (B_{modify}). Operations on the whole file are similar to those on a data block, which will no longer be described here.

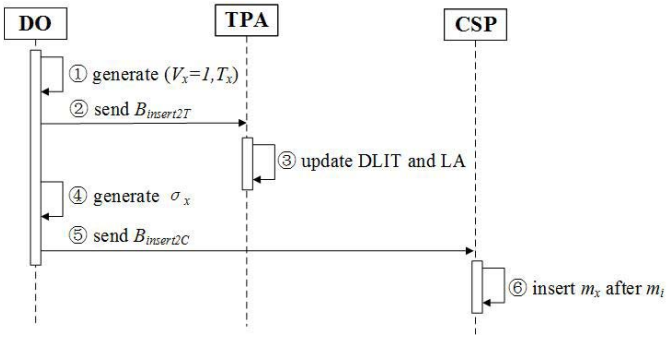

Fig. 9. Block insertion: insert m_x after m_i.

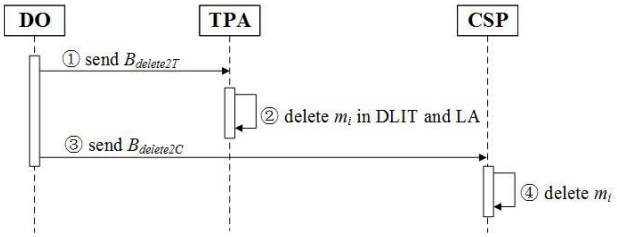

Fig. 10. Block deletion: delete m_i.

1) Block Insertion: We suppose that a new data block m_x will be inserted after m_i in file F_{ID} of owner U_{ID}, as shown in Fig. 9. First, the DO generates the corresponding version number and time stamp (V_x, T_x) for m_x to be inserted into the DLIT, where $V_x = 1$ because it is a new block. Then, the DO sends the insertion instruction $B_{insert2T} = (insert, U_{ID}, F_{ID}, i, x, V_x, T_x)$ to the TPA. Upon receiving, the TPA updates the records in the DLIT as well as the LA. The change in the DLIT can be found in Fig. 3, which simply differs in action objects. In addition, the TPA can be found in Fig. 8(a) and 8(b) for operating on the LA. Meanwhile, the DO generates the corresponding authenticator σ_x for m_x, sending $B_{insert2C} = (insert, U_{ID}, F_{ID}, i, m_x, \sigma_x)$ to the CSP. Once receiving, the CSP inserts a corresponding file block after m_i.

2) Block Deletion: As shown in Fig. 10, we suppose that block m_i will be deleted from file F_{ID}, which belongs to U_{ID}. Similar to insertion but much easier, data along with their information would be updated as well. The DO would no longer generate new parameters for the TPA or the CSP this time. The DO sends $B_{delete2T} = (delete, U_{ID}, F_{ID}, i)$ to the TPA. Upon receipt, the TPA would find m_i and delete its information in the DLIT and LA. In addition, the DO sends $B_{delete2C} = (delete, U_{ID}, F_{ID}, i)$ to the CSP, who executes the deletion instruction as directed.

3) Block Modification: We suppose that the block m_i will be modified to m_i* for some reason, shown in Fig. 11. First, the DO generates the corresponding version number and time stamp (V_i*, T_i*) for it, where $V_i* = V_i + 1$ and the time stamp is independent of the previous time stamp. Then, the DO sends $B_{modify2T} = (modify, U_{ID}, F_{ID}, i, V_i*, T_i*)$ to the TPA. Upon receipt, the TPA only updates the DLIT because the sequence of blocks remains unchanged. Simultaneously, the DO generates a new authenticator σ_i* for the block,

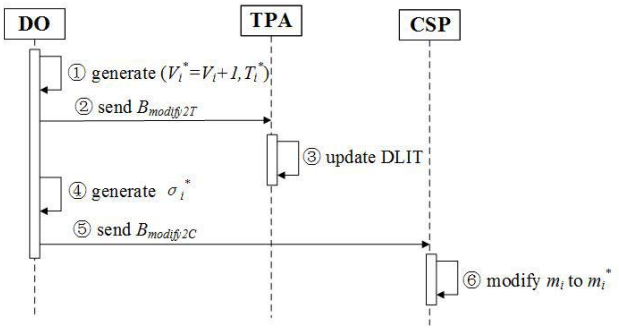

Fig. 11. Block modification: modify m_i to $m_i{}^*$.

sending $B_{modify2C} = (modify, U_{ID}, F_{ID}, i, m_i{}^*, \sigma_i{}^*)$ to the CSP for updating. Once received, the CSP modifies m_i as indicated.

Note that data freshness is always a tradeoff of both time and energy. Hence, lazy update is practical for the scenario where the DO prefers to reduce online overhead or some other aspect. A queue could be employed to realize lazy update, which is equipped with the first in, first out property. In this way, dynamic operations on cloud data can be executed in an orderly manner rather than at random.

VI. Evaluation

In this section, some analysis concerning the proposed auditing protocol will be presented, including correctness analysis, security analysis and performance analysis.

A. Correctness Analysis

There are three kinds of verification processes in our auditing protocol: a single auditing task from a single DO, multiple auditing tasks from a single DO and multiple auditing tasks from multiple DOs.

For each challenge and the corresponding valid proof, the verification equations hold, while for each challenge and the invalid proof, the verification equations fail. In another word, once the proof is responded from the CSP, the validity of the proof can be judged through the verification equations correctly. Therefore, in this part, similar to the correctness analysis in [24], [26], [27], [43], and [44], we will demonstrate the correctness of the designed auditing protocol by proving the equalities of the verification equations for the three kinds of verification processes mentioned above. Specifically, the correctness of Eq. (7), Eq. (8), and Eq. (9) will be elaborated separately.

Eq. (7) can verify a single auditing task from a single DO. The correctness of Eq. (7) is elaborated as follows:

$$\begin{aligned} e(T,g) &= e(\prod_{i\in[1,s]} \sigma_i^{r_i}, g) = e(\prod_{i\in[1,s]} (h(V_i||T_i)\cdot u^{m_i})^{a\cdot r_i}, g) \\ &= e(\prod_{i\in[1,s]} (h(V_i||T_i)\cdot u^{m_i\cdot r_i}), g^a) \\ &= e(\prod_{i\in[1,s]} (h(V_i||T_i)\cdot u^{m_i\cdot r_i}), v) \\ &= e(\prod_{i\in[1,s]} (h(V_i||T_i))\cdot u^{\sum_{i\in[1,s]} m_i\cdot r_i}, v) \\ &= e(\prod_{i\in[1,s]} h(V_i||T_i), v)\cdot e(u^{\sum_{i\in[1,s]} m_i\cdot r_i}, v) \\ &= \prod_{i\in[1,s]} DI_i \cdot e(u^D, v) \\ &= DI\cdot e(u^D, v) \end{aligned} \tag{10}$$

Eq. (8) can verify multiple auditing tasks from a single DO. The correctness of Eq. (8) is demonstrated as follows:

$$\begin{aligned} \prod_{f_i\in[1,f_s]} e(T,g) &= \prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} \sigma_{f_i i}{}^{r_{f_i i}}, g) \\ &= e(\prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} (h(V_{f_i i}||T_{f_i i})\cdot u^{m_{f_i i}})^{a r_{f_i i}}, g) \\ &= e(\prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} (h(V_{f_i i}||T_{f_i i})\cdot u^{m_{f_i i}\cdot r_{f_i i}}), v) \\ &= \prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} (h(V_{f_i i}||T_{f_i i})) \\ &\quad \cdot u^{\sum_{i\in[1,s]} m_{f_i i}\cdot r_{f_i i}}, v) \\ &= \prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} (h(V_{f_i i}||T_{f_i i})), v) \\ &\quad \cdot \prod_{f_i\in[1,f_s]} e(u^{\sum_{i\in[1,s]} m_{f_i i}\cdot r_{f_i i}}, v) \\ &= \prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} DI_i \cdot \prod_{f_i\in[1,f_s]} e(u^{D_{f_i}}, v) \\ &= \prod_{f_i\in[1,f_s]} DI\cdot e(u^D, v) \end{aligned} \tag{11}$$

Eq. (9) can verify multiple auditing tasks from multiple DOs. The correctness of Eq. (9) is proved as follows:

$$\begin{aligned} &\prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(T,g) \\ &= \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} \sigma_{u_i f_i i}{}^{r_{u_i f_i i}}, g) \\ &= e(\prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} \\ &\quad \times (h(V_{u_i f_i i}||T_{u_i f_i i})\cdot u^{m_{u_i f_i i}})^{a\cdot r_{u_i f_i i}}, g) \\ &= e(\prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} (h(V_{u_i f_i i}||T_{u_i f_i i}) \\ &\quad \cdot u^{m_{u_i f_i i}\cdot r_{u_i f_i i}}), v) \\ &= \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} (h(V_{u_i f_i i}||T_{u_i f_i i})) \\ &\quad \cdot u^{\sum_{i\in[1,s]} m_{u_i f_i i}\cdot r_{u_i f_i i}}, v) \\ &= \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(\prod_{i\in[1,s]} (h(V_{u_i f_i i}||T_{u_i f_i i})), v) \\ &\quad \cdot \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(u^{\sum_{i\in[1,s]} m_{u_i f_i i}\cdot r_{u_i f_i i}}, v) \\ &= \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]}\prod_{i\in[1,s]} DI_i \cdot \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} e(u^{D_{u_i f_i}}, v) \\ &= \prod_{u_i\in[1,u_s]}\prod_{f_i\in[1,f_s]} DI\cdot e(u^D, v) \end{aligned} \tag{12}$$

B. Security Analysis

We will prove the security of our protocol with proofs of the following several theorems.

Theorem 1: The BLS-based Homomorphic Verifiable Authenticator used in our protocol cannot be forged by any adversary if the Computational Differ-Hellman assumption in bilinear group holds.

Proof: This theorem suggests that forging an BLS-HVA is computationally infeasible. As shown in the security analysis of Wang *et al.* [30], the HVA scheme has been proved to be effectively unforgeable, the reason for which is that BLS is secure when the CDH problem is difficult in bilinear groups [38]. The proof can be found in [30] and is omitted here. □

Theorem 2: During the verification process, replay attacks would never be effective. In other words, a malicious CSP could not pass verification if it responds to the challenge from the TPA with previous information.

Proof: We define the replay attack game as follows, which occurs in the verify phase. The DO asks the TPA to check a certain file in the cloud. Upon receipt of such a delegation, the TPA picks random blocks for verification and sends the challenge $chal = \{i, r_i\}_{i\in[1,s]}$ to the CSP. However, the CSP is currently experiencing a replay attack. Hence, a part of the response from the CSP would be replaced by previous data information. We assume that the j-th block's information is replaced by its former block m_{j^-}. In other words, the CSP generates the proof (T^-, D^-) as a response to the challenge from the TPA, where the tag proof

$$T^- = \prod_{i\in[1,j-1]\bigcup[j+1,s]} {\sigma_i}^{r_i} \cdot {\sigma_{j^-}}^{r_{j^-}}, \tag{13}$$

and the data proof

$$D^- = \sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i \cdot r_i + m_{j^-} \cdot r_{j^-}. \tag{14}$$

Upon receipt, the TPA checks the validity of the proof. If the above proof passes verification, the CSP experiencing replay attacks wins the game; otherwise, it does not.

To be substantially more convincing, we will analyze Eq. (7) to prove the impossibility of replay attacks. The left part concerning the tag proof replaced by previous information can be written as follows:

$$\begin{aligned} e(T^-, g) &= e(\prod_{i\in[1,j-1]\bigcup[j+1,s]} {\sigma_i}^{r_i} \cdot {\sigma_{j^-}}^{r_{j^-}}, g) \\ &= e(\prod_{i\in[1,j-1]\bigcup[j+1,s]} (h(V_i||T_i) \cdot u^{m_i})^{a\cdot r_i} \\ &\quad \cdot (h(V_{j^-}||T_{j^-}) \cdot u^{m_{j^-}})^{a\cdot r_{j^-}}, g) \\ &= e(\prod_{i\in[1,j-1]\bigcup[j+1,s]} (h(V_i||T_i) \cdot u^{m_i})^{r_i} \\ &\quad \cdot (h(V_{j^-}||T_{j^-}) \cdot u^{m_{j^-}})^{r_{j^-}}, v) \\ &= e(\prod_{i\in[1,j-1]\bigcup[j+1,s]} (h(V_i||T_i)) \cdot h(V_{j^-}||T_{j^-}) \\ &\quad \cdot u^{\sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i\cdot r_i + m_{j^-}\cdot r_{j^-}}, v) \end{aligned} \tag{15}$$

The right part concerning the data proof can be written as follows:

$$\begin{aligned} DI \cdot e(u^{D^-}, v) &= \prod_{i\in[1,s]} DI_i \cdot e(u^{D^-}, v) \\ &= e(\prod_{i\in[1,s]} h(V_i||T_i), v) \\ &\quad \cdot e(u^{\sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i\cdot r_i + m_{j^-}\cdot r_{j^-}}, v) \\ &= e(\prod_{i\in[1,s]} (h(V_i||T_i)) \\ &\quad \cdot u^{\sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i\cdot r_i + m_{j^-}\cdot r_{j^-}}, v) \\ &= e(\prod_{i\in[1,j-1]\bigcup[j+1,s]} (h(V_i||T_i)) \cdot h(V_j||T_j) \\ &\quad \cdot u^{\sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i\cdot r_i + m_{j^-}\cdot r_{j^-}}, v) \end{aligned} \tag{16}$$

Comparing the above two equations, we can believe that only when $h(V_{j^-}\|T_{j^-}) = h(V_j\|T_j)$, i.e., $V_{j^-} = V_j$ and $T_{j^-} = T_j$, can the results of the two equations be equal. To the best of our knowledge, however, their version number and time stamp can never be the same. Hence, the results of Eq. (15) and (16) can never equal each other. In other words, replay attacks can never succeed under our protocol. □

Theorem 3: During the verification process, replacing attacks would also never work. In other words, if the CSP corrupts or discards a challenged block, it is impossible for the CSP to succeed in replacing such a data block with another block to respond to the challenge to maintain its reputation.

Proof: We define the replacing attack game as follows, similar to the replay attack above. The DO once again delegates to the TPA an auditing task. After receipt, the TPA randomly chooses several data blocks of the indicated file and generates the challenge $chal = \{i, r_i\}_{i\in[1,s]}$. In this scenario, the CSP is experiencing a replacing attack at this moment. We assume that m_j along with its information is replaced with m_r. Therefore, the proof response from the CSP would be (T^r, D^r), where the tag proof is

$$T^r = \prod_{i\in[1,j-1]\bigcup[j+1,s]} {\sigma_i}^{r_i} \cdot {\sigma_r}^{r_r}, \tag{17}$$

and data proof is

$$D^- = \sum_{i\in[1,j-1]\bigcup[j+1,s]} m_i \cdot r_i + m_r \cdot r_r. \tag{18}$$

A convincing derivation of the verification equation with T^r and D^r plugged in is omitted here but is similar to the analysis of the impossibility of replay attacks. □

C. Performance Analysis

In this subsection, we will introduce the performance analysis of the designed auditing protocol and compare it with the state of the art using numerical results.

1) Communication Cost Analysis: The communication cost of the proposed protocol is analyzed and compared in this section. As mentioned above, the protocol consists of a setup phase and a verify phase, of which only the verify phase introduces communication costs. The TPA should send a challenge to the CSP for auditing, producing a communication

TABLE II
COMMUNICATION COST COMPARISON

Auditing protocols	Communication cost of verification
Ref. [29]	$O(slogn)$
Ref. [30]	$O(slogn)$
Ref. [31]	$O(s)$
Ref. [17]	$O(s)$
Our protocol	$O(s)$

overhead of $O(s)$, where s is the number of challenged blocks, as denoted in the detailed protocol. Then, the CSP returns the proof, which brings in $O(1)$ communication cost. Finally, the TPA informs the DO of the auditing result, which costs $O(1)$ as well. Overall, the total communication cost is $O(s)$.

Table II presents comparison results of our protocol with several popular auditing protocols mentioned in the related works, where n denotes the total number of data blocks in a outsourced file and s represents the number of challenged blocks. Clearly, our protocol achieves a high level of performance; moreover, it is substantially more efficient than the other protocols.

2) Computational Cost Analysis: The computational cost of our protocol is presented in this section. Effectively, our protocol requires almost the same communication costs as [31] and [17], as compared in Table II, which are less than those of the other two protocols. Therefore, we only compare the computational overhead of our protocol with [17] and [31], as shown in Table III. Briefly, we denote point multiplication as M, modular exponentiation as E, and the Pairing operation as P. The hash operation is ignored here because of its negligible cost compared with E, M, and P. In the table, n is the total number of data blocks outsourced, s is the number of challenged blocks, and m is the number of sectors in a given data block.

In the setup phase, the computational cost is mainly concentrated on the DO side, which includes key generation and file pre-processing. The cost is $1E$ for KeyGen, which is the same as [17] and $1E$ less than in [31]. In addition, the cost is $2nE + nM$ for Filepro2C, which is $nE + 2mnM$ less than [31]. Note that, in [17], $nP+nsE+nsM$ with $*$ marked is outsourced to the cloud, and the DO side only costs $nE+nM$, which is nE less than in our protocol. However, it is well known that a cloud is a pay-as-you-go service system; in other words, computing outsourcing is not free to us. Hence, it is not necessary to trade off so much money for outsourcing $nP + nsE + nsM$ to save $1/(1 + 2s)$ of the outsourced computing. For ProofGen, the cost is $sE + (s - 1)M$ for tag proof generation, and the cost is sM for data proof generation. The total cost is significantly less than in [17] and [31].

In addition, some simulation experiments have been conducted to prove the efficiency of our protocol. The experiments were performed on a desktop running Windows 7 with a 2.50 GHz Intel Core i5-2450M CPU and 12 GB of memory. The codes were written in MyEclipse with the Java language to obtain the data results, which are the results of 50 runs to obtain the averages as the final data. In addition, the figures are drawn in MATLAB 2014 with data obtained from MyEclipse 2014. As shown in Table III, it is obvious that [31] costs substantially more than our protocol, which almost loses its meaning in comparison with our protocol. Hence, we only compare our protocol with [17].

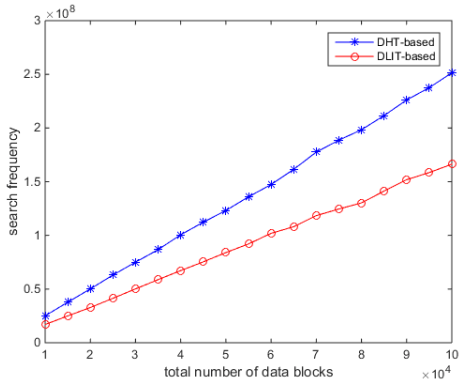

Fig. 12. Search frequency comparison under constant challenged blocks.

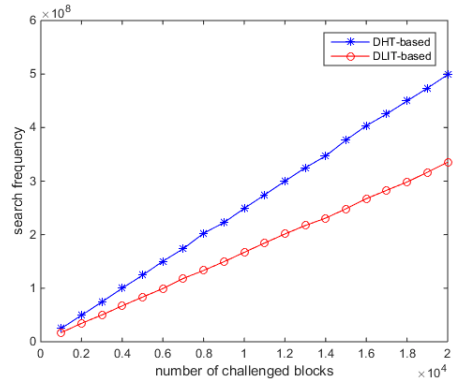

Fig. 13. Search frequency comparison under constant total number of blocks.

First, experiments are performed on a single file, of which the results are shown in Fig. 12 and Fig. 13. We divide the comparison scenario into two categories: a scenario under a changing total number of blocks from 1×10^4 to 1×10^5 with a constant number of challenged blocks set as 5000 and a scenario under a changing number of challenged blocks from 1000 to 2×10^4 with a constant total number of blocks set as 5×10^4. Fig. 12 is under the first scenario, where the blue line with stars represents he DHT-based protocol in [17] and the red line with circles represents our DLIT-based protocol. The search frequency of given challenged blocks is plotted on the Y-axis against the total number of blocks on the X-axis. It is clear that the DLIT-based protocol always costs less to locate the same number of challenged blocks. Moreover, the greater the total number of blocks, the greater the frequency gap. Fig. 13 indicates the other scenario, where the blue line with stars and the red line with circles denote the same two protocols as Fig. 12. The search frequency of the given number of challenged blocks remains on the Y-axis, and the number of challenged blocks is on the X-axis. Obviously, the DLIT-based protocol is better than the DHT-based protocol because the latter requires a greater frequency to search the same number of blocks.

TABLE III
COMPUTATIONAL COST COMPARISON

Protocol	KeyGen	File Pre-processing	ProofGen
Ref. [31]	$2E$	$3nE + n(2m+1)M$	$sE + (ms + 2s + m - 1)M$
Ref. [17]	$1E$	$(nE + nM) + (nP + nsE + nsM)^*$	$(s+1)E + (2s-1)M + 1P$
Our protocol	$1E$	$2nE + nM$	$sE + (2s-1)M$

† E : Modular Exponentiation ; M : Point Multiplication; P : Bilinear Pairing.
† n : The total number of data blocks outsourced.
† s : The number of challenged blocks.
† m : The number of sectors in a given data block.
† * : The computational cost of file pre-processing outsourced to the cloud.

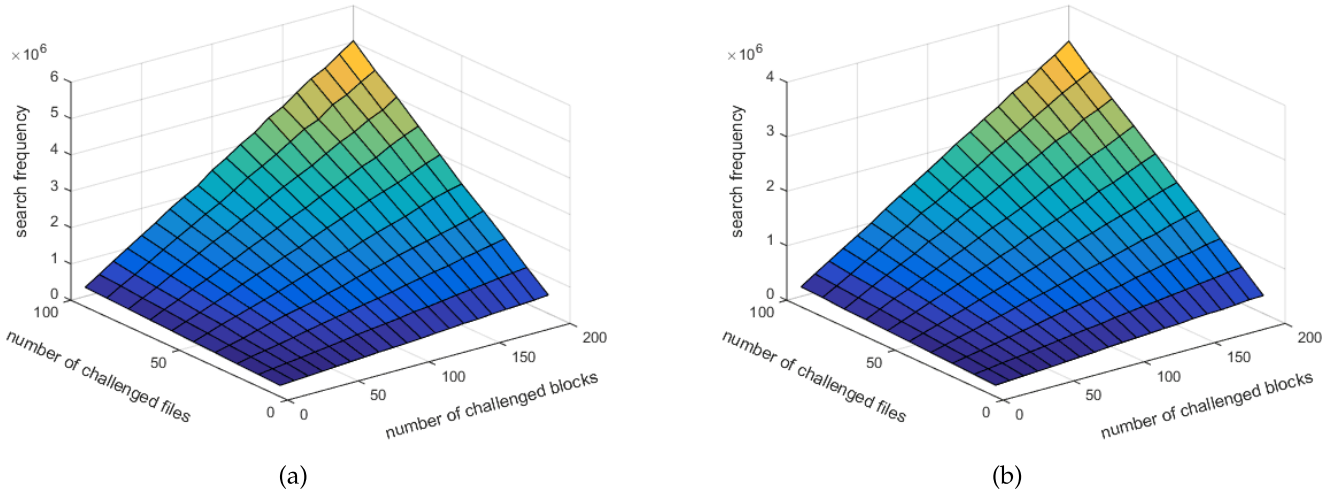

Fig. 14. Search frequency comparison of batch auditing. (a) Batch auditing under the DHT-based protocol. (b) Batch auditing under the DLIT-based protocol.

Then, we compare the efficiency of batch auditing. We set the total block number of each file to 500, the number of challenged files varies from 10 to 100, and the number of challenged blocks of each file varies from 10 to 200. As shown in Fig. 14, (a) is the experimental result of the DHT-based protocol in [17] and (b) is the result of our DLIT-based protocol. The search frequency increases when either the number of challenged files or the number of blocks increases. It is clear that the upward tendencies of the two figures are similar, although the tendency of our protocol is much gentler. For instance, when we challenge 100 files with 200 random blocks in each file, the total number of random blocks being 100×200, our protocol needs to perform 3.35×10^6 searches, whereas the DHT-based protocol needs to perform 5.04×10^6 searches. Clearly, the protocol in [17] is almost 1.5-times more computationally expensive than our protocol, which indicates the efficiency of our batch auditing.

In summary, the auditing protocol proposed in this paper performs better and is less computationally expensive compared to the state of the art.

VII. CONCLUSION

In this paper, we propose a public auditing protocol with a novel dynamic structure composed of a *doubly linked info table* and a *location array*. Compared with the state of the art, an appropriate relationship between the DLIT and the LA makes our protocol perform better both in terms of efficient dynamic support and reduced overhead. Moreover, some basic challenges in cloud auditing, such as batch auditing, blockless verification and lazy update, have been overcome by our protocol. Sufficient theoretical proof indicates the security of our protocol. Extensive numerical analysis and experimental comparison results could be used to validate the performance of our protocol, making it substantially more convincing.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Nat. Inst. Standards Technol.*, vol. 53, no. 6, p. 50, 2011.
[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generat. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.
[3] K. Yang and X. Jia, "Data storage auditing service in cloud computing: Challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.
[4] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
[5] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Trans. Commun.*, vol. 98, no. 1, pp. 190–200, 2015.
[6] J. Shen, H. Tan, S. Moh, I. Chung, Q. Liu, and X. Sun, "Enhanced secure sensor association and key management in wireless body area networks," *J. Commun. Netw.*, vol. 17, no. 5, pp. 453–462, 2015.

[7] M. Green, "The threat in the cloud," *IEEE Security Privacy*, vol. 11, no. 1, pp. 86–89, Jan./Feb. 2013.

[8] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Syst. J.*, to be published.

[9] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: A survey," *Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 113–170, Apr. 2014.

[10] L. F. B. Soares, D. A. B. Fernandes, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, *Cloud Security: State of the Art*. Berlin, Germany: Springer, 2014.

[11] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.

[12] Q. Jiang, M. K. Khan, X. Lu, J. Ma, and D. He, "A privacy preserving three-factor authentication protocol for e-health clouds," *J. Supercomput.*, vol. 72, no. 10, pp. 3826–3849, 2016.

[13] E. B. Dudin and Y. G. Smetanin, "A review of cloud computing," *Sci. Tech. Inf. Process.*, vol. 38, no. 4, pp. 280–284, 2011.

[14] J. Shen, H. Tan, J. Wang, J. Wang, and S. Lee, "A novel routing protocol providing good transmission reliability in underwater sensor networks," *J. Internet Technol.*, vol. 16, no. 1, pp. 171–178, 2015.

[15] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[16] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 5, pp. 546–556, Sep. 2015.

[17] H. Tian *et al.*, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Serv. Comput.*, to be published.

[18] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[19] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," *J. Internet Technol.*, vol. 16, no. 2, pp. 317–323, 2015.

[20] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[21] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive Report 186, HP Lab No. Tech. Rep. HPL-2008-32, Apr. 2008, pp. 477–494.

[22] F. Sebe, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.

[23] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[24] C. Wang, S. 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, Feb. 2013.

[25] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Comput. Electr. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2014.

[26] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[27] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[28] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks*, 2008, pp. 1–10.

[29] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, pp. 213–222, 2009.

[30] 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, May 2011.

[31] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 227–238, Apr./Jun. 2013.

[32] H. Jin, H. Jiang, and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Trans. Cloud Comput.*, to be published.

[33] J. Shen, W. Zheng, J. Wang, Y. Zheng, X. Sun, and S. Lee, "An efficient verifiably encrypted signature from weil pairing," *J. Internet Technol.*, vol. 14, no. 6, pp. 947–952, 2013.

[34] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 213–229, 2015.

[35] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Syst. J.*, to be published.

[36] J. Shen, S. Moh, and I. Chung, "Identity-based key agreement protocol employing a symmetric balanced incomplete block design," *J. Commun. Netw.*, vol. 14, no. 6, pp. 682–691, Dec. 2012.

[37] X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, and K. Kim, "Identity-based chameleon hashing and signatures without key exposure," *Inf. Sci.*, vol. 265, no. 5, pp. 198–210, 2014.

[38] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.

[39] J. Kar, "Provably secure identity-based aggregate signature scheme," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, 2012, pp. 137–142.

[40] S. S. D. Selvi, S. S. Vivek, J. Shriram, and C. P. Rangan, "Identity based partial aggregate signature scheme without pairing," in *Proc. SARNOFF*, May 2012, pp. 1–6.

[41] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[42] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "Restricted universal designated verifier signature," in *Ubiquitous Intelligence and Computing* (Lecture Notes in Computer Science), vol. 4159. Berlin, Germany: Springer-Verlag, 2006, pp. 874–882.

[43] 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, Aug. 2015.

[44] J. Liu, K. Huang, H. Rong, and H. Wang, "Privacy-preserving public auditing for regenerating-code-based cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1513–1528, Jul. 2015.

Jian Shen received the M.E. and Ph.D. degrees in computer science from Chosun University, South Korea, in 2009 and 2012, respectively. Since 2012, he has been a Professor with the Nanjing University of Information Science and Technology, Nanjing, China. His current research interests include public key cryptography, secure data sharing, and data auditing in cloud.

Jun Shen received the B.S. degree in 2015. She is currently pursuing the M.E. degree with the Nanjing University of Information Science and Technology, Nanjing, China. She focuses on the security and privacy issues in cloud environment. Her current research interests are security and privacy in cloud, and auditing protocols for cloud storage.

Xiaofeng Chen (SM'16) received the B.S. and M.S. degrees in mathematics from Northwest University, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University in 2003. He is currently with Xidian University as a Professor. He has authored or co-authored over 100 research papers in refereed international conferences and journals. His current research interests include applied cryptography and cloud computing security. His work has been cited over 3000 times in Google Scholar. He is on the Editorial Board of *Security and Communication Networks*, *Computing and Informatics*, and the *International Journal of Embedded Systems*. He has served as the Program/ General Chair or Program Committee Member in over 30 international conferences.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. He has authored or co-authored over 60 research papers in refereed international conferences and journals. His current research interests include cryptography and information security. His work has been cited over 1000 times in Google Scholar. He is on the Editorial Board of the *International Journal of Information Security* (Springer) and has served as the Program/General Chair or Program Committee Member in over 40 international conferences.

Willy Susilo (SM'01) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology and the Director of Institute of Cybersecurity and Cryptology with the University of Wollongong. He has authored or co-authored over 300 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. His work has been cited over 9000 times in Google Scholar. He has served as a Program Committee Member in dozens of international conferences. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council and the Researcher of the Year award in 2016 by the University of Wollongong. He is Editor-in-Chief of the *Information* journal. He is currently serving as an Associate Editor for several international journals, including *Elsevier Computer Standards and Interface* and the *International Journal of Information Security* (Springer).