Received October 4, 2019, accepted October 24, 2019, date of publication November 1, 2019, date of current version November 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950938

Fuzzy Identity-Based Dynamic Auditing of Big Data on Cloud Storage

CHENBIN ZHAO[1,2], LI XU[1,2], (Member, IEEE), JIGUO LI[1,2], FENG WANG[1,2], AND HE FANG[3], (Member, IEEE)

[1]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, China
[2]Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China
[3]Department of Electrical and Computer Engineering, Western University, London, ON N6A 3K7, Canada

Corresponding author: Li Xu (xuli@fjnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61771140, Grant 61972095, and Grant U1736112, in part by the Major Science and Technology Project in Fujian Province under Grant 2017H6005, in part by the Major Program of Fuzhou Municipal Bureau of Science and Technology under Grant (2017)325, and in part by the Enterprise Cooperation Project under Grant DH-1307.

ABSTRACT To ensure the reliability and integrity of data in the cloud storage server, some scholars provided various data integrity auditing schemes. However, the most existing data integrity auditing schemes only support the static data and may be unsuitable for the dynamic operations of data. To overcome this difficulty, we propose a fuzzy identity-based dynamic auditing of big data, which combines the structure of the Merkle hash tree (MHT) with the Index logic table (ILT). Our scheme not only performs the dynamic operations of data block in the ILT, namely modification, insertion and deletion, but also efficiently executes dynamic operations of the ILT on the structure of the MHT. We also elaborate the security, characteristics and performance analysis of the proposed scheme separately. The analysis results show that the proposed scheme costs less time than the structure of the original MHT to generate the root node hash value during the metadata generation phase and update the root node hash value during the dynamic operations. Furthermore, when users store the new ILT in local storage, they require lower communication cost to update root node hash value than users without storing the ILT, and fewer interactions between the cloud storage server and users in the dynamic operations process.

INDEX TERMS Big data, cloud storage, data integrity, dynamic operation, fuzzy identity-based.

I. INTRODUCTION

The development of the big data era has brought tremendous pressure on storage capacity and cost [1]. Users have permission to upload data to the cloud storage server. Consequently, the cloud storage server reduces the burden of users on data storage management and maintenance, as well as avoiding expensive infrastructure costs. It gradually becomes a popular platform for the customer to outsource data, because of its large storage capacity, whenever and wherever accessing to resources, easy management and other typical advantages. Unfortunately, it does pose numerous security challenges. Although, other techniques had been well studied, as exemplified by the physical layer security [2], [3], which utilized the specific attributes of communication links, it suffered from the unreliable performance because of the physical layer attributes used. Not only there are some problems in physical layer technology, there are no doubt that cloud storage server also faces many security problems and challenges. Cloud storage server is not completely trusted platform. When data is uploaded to the cloud storage server, the data owners lose the right to control their data. The authors proposed various solutions [4]–[7] for secure access control of data. In a cloud-based medical system, images data should be encrypted prior to being outsourced [8]. The rise of Internet of Things technology and cloud computing, it is also very crucial to ensure data security and efficiently search in the ciphertext status [9] [10]. Regrettably, there also were many data loss or leakage incidents in recent years. In August 2015, due to thunder and lightning weather, the disk of Google's data center was damaged and data information was lost permanently. Before long sales forces lost their customer data due to internal staff misconducts. The cloud storage server maintains its reputation by hiding data loss incidents.

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

Even worse, the cloud storage server deletes data that is not frequently accessed in order to save storage spaces and benefits from it. In summary, when users upload their data to the cloud storage server, they exactly know whether their data is preserved perfectly. Therefore, it is very crucial for users to guarantee that data is intact without damage.

A. MOTIVATION

Firstly, some scholars put forward various auditing schemes [11]–[13] for the purpose of ensuring the integrity, which adopted the public key infrastructure (PKI). In the auditing process, the auditor needs to apply for the user's public key certificate and extracts the public key to complete the auditing. As we all know, each user needs to apply for a certificate in advance and the application process of certificate is considerably complex. Besides, the certificate management, revocation as well as other processes increase the burden of the system. In order to solve above issues, the authors proposed identity-based and attribute-based cryptosystems [4]–[7], [14], [15]. In Li *et al.* [14] proposed a fuzzy identity-based data auditing scheme, but did not execute the dynamic operations. We consider adopting the fuzzy identity-based cryptosystem and reconstruct a new scheme with dynamic operations. Users get the corresponding private key by inputting the identity set. Subsequently, the data is signed by the private key to ensure the integrity.

Secondly, the previous auditing schemes [11], [14] only aimed at static data, and did not support the dynamic operations. In [13], although Wang et al. provided the scheme with the dynamic operations, which combined with dynamic structure of the MHT. On the one hand, this scheme was based on the PKI system. On the other hand, when plenty of data blocks were inserted, the structure of MHT was gradually enlarging. Therefore, we consider combining the structure of MHT with the ILT [15], and use the ILT to replace the leaf node of MHT. When the ILT is full by inserting data blocks, users insert the new ILT. Hence, our scheme is suitable for big data scenarios. Sometimes, when the ILT is empty by deleting the data blocks, users delete the leaf node in order to delete the empty ILT. In addition, we assume that users temporarily store the new ILT in local storage. After the dynamic operations are completed, users update the new ILT and the according root node. For instance, university teachers input grades of students at the end of the semester. After grades of all students in the class are uploaded, teachers update data information of all students again. In other words, dynamic operations are controlled by users themselves.

B. CONTRIBUTIONS

To overcome the exiting problems, we design a fuzzy identity-based dynamic auditing scheme of big data on cloud storage. The main contributions are summarized as follows:

- We first put forward a fuzzy identity-based dynamic data auditing scheme. The scheme avoids the complex public key certificate management problems. In addition, when the number of identity set reaches the threshold value, users are allowed to perform dynamic operations. Compared with the exiting auditing schemes, dynamic operations of the proposed scheme are more flexible.
- We combine the dynamic structure of MHT with the ILT. The scheme supports two categories dynamic operations, namely data block dynamic operations and ILT dynamic operations. Each leaf node is replaced by the ILT and each ILT stores numerous data blocks. This also makes our solution more suitable for big data scenarios.
- Analysis and comparison results show that the computation cost of generating root node is lower than the structure of the original MHT in metadata generation phase and dynamic operations phase. If the users do not store the new ILT in local storage, dynamic operations for data blocks lead to the update of the root node. Thus, the communication overhead of users is also increased in the dynamic operations phase. Users store the new ILT in our scheme and control the update of the root node by themselves. Analysis results show that not only the communication cost of users is lower than users who do not store ILT, but also the interactions between the cloud storage server and users are also fewer.

II. RELATED WORK

Deswarte *et al.* [16] utilized the RSA hash function to propose a remote data integrity scheme. However, the computation overhead is very high. Ateniese *et al.* [11] presented a provable data possession (PDP) scheme after plenty of researches, which used sampling technology to verify data integrity and effectively ensured data security. In 2007, Juels and Kaliski [17] proposed a proof of retrievability (PoR) scheme. It not only verified the integrity of remote data, but also utilized error-correcting codes to recover data that was damaged with a certain probability. However, PoR scheme did not support any effective extensions, and none of the previous solutions supported dynamic operations.

In 2009, Erway *et al.* [12] provided a formal framework for dynamic verifiable data possession by extending the PDP model. They constructed two fully dynamic PDP protocols by utilizing the RSA tree structure and the hierarchical authentication jump table, respectively. Regrettably, the computation efficiency did not have advantages. In Sebe *et al.* [18] proposed an auditing protocol, which was easy to extend to support dynamic operations but did not support public verification. Thus, verification between the cloud storage server and users did not guarantee fairness and impartiality. In 2011, Wang *et al.* [13] provided a scheme on the basis of the BLS, which combined the structure of MHT to support fully dynamic operations. They also utilized the TPA to achieve the public verification. In Zhang and Blanton [19] presented a new authentication data structure, namely balanced update tree, which each node stored a certain number of data blocks labels. In the auditing process, the cloud storage server generated the responses, which contained all labels at each node. Therefore, communication overhead was increased dramatically. With the continuous development of researches, various extended auditing schemes such as certificateless public integrity checking scheme [20], remote data possession checking protocol [21], [22], cross-cloud platform auditing scheme [23]–[25] were proposed to meet multifarious requirements.

However, most schemes above described on the basis of the PKI system. Afterward, a host of identity-based data integrity auditing schemes were proposed. In Zhang and Dong [26] presented an identity-based integrity auditing scheme, which supported public verification and privacy-preserving. Yu *et al.* [27] provided a provably secure identity-based auditing scheme, and Li *et al.* [14] put forward a fuzzy identity-based data auditing scheme. Although the above schemes avoided the complex management of certificate, they only considered static data and did not execute dynamic operations. Therefore, the above schemes described were not suitable for the cloud storage environment.

Organization: The rest of this article is arranged as below. In Section III, we describe some preliminaries. In Section IV, the system model and security model for our scheme are introduced. In Section V, we give a detailed description of the proposed scheme. Security analysis is provided in Section VI and characteristic comparisons and performance analysis are described in Section VII. Finally, we give a conclusion of the paper and the future research work in Section VIII.

III. PRELIMINARIES

In this section, bilinear pairing, threshold secret sharing (TSS), merkle hash tree (MHT), index logic table (ILT), and fuzzy identity-based signature are illustrated. In TABLE 1, we give some descriptions of notations in this paper.

TABLE 1. The descriptions of notations.

Notations	Descriptions
k	The security parameter
e	The bilinear pairing map
G_1, G_2	Two multiplicative cyclic groups
Θ	The auxiliary authentication information (AAI)
d	The error tolerance value
ω	The identity set of the user
MHT/ ILT	Merkle hash tree/ Index logic table
pp	The system public parameters
mk	The master key
$\widehat{SK}_\omega$	The secret key of the cloud user
Φ	The data block tag
Γ	The index logic table tag
$Chal$	The challenge information
$resp$	The response information

A. BILINEAR PAIRING

G_1 and G_2 are two multiplicative cyclic groups with the prime order q, and g is a generator in G_1, A bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$ is a map satisfying the following properties [28]:

- *Bilinear:* $e(g^a, g^b) = e(g, g)^{ab}$, for $g \in G_1$ and $a, b \in Z_q$.
- *Non-degenerate:* $e(g, g) \neq 1$.
- *Computational:* $e(u, v)$ is computable for any $u, v \in G_1$.

B. THRESHOLD SECRET SHARING (TSS)

In [29], Shamir proposed the secret sharing scheme. The secret value s is divided into several parts to each participant in the group, and those parts are called shares. The secret value s can be restored with plenty of shares. Concretely, we assume that $\rho = (p_1, p_2, \ldots, p_n)$ denote n participates, a (k, n) TSS scheme means that the set of n participates, any k members' shares or more participants reconstruct the secret value s but less than k participants do not recover, where k is the threshold value. On the basis of the polynomial interpolation, there is a polynomial of degree $k-1$ for taking k points. Assume that a secret value s is shared. The dealer picks $k-1$ integers $a_1, a_2, \ldots, a_{k-1}$ randomly and defines a polynomial $f(x)$ of degree $k-1$, which is described as $f(x) = s + \sum_{i=0}^{k-1} a_i \cdot x^i$ and $f(0) = s$. Then, the dealer randomly selects some $i \in Z_p$ for each p_i, and defines the secret share $s_i = f(i)$. If a set of participants $S \subset \rho$, where $|S| \geqslant k$, they reconstruct the secret value s and also recover the $f(x)$ by $f(x) = \sum_{P_i \in S} \Delta_{i,s}(x) \cdot s_i$, where $\Delta_{i,s}(x) = \prod_{P_i \in S, j \neq i} \frac{x-j}{i-j}$ is the lagrange coefficient.

C. MERKLE HASH TREE (MHT)

MHT [13] is a data structure for research authentication, which proves that elements are efficient and secure and not damaged or modified. It is composed of a binary tree and the hash values of the authentication data, which are utilized for the leaf nodes of the MHT. FIGURE 1 describes a simple instance of authentication. $h(\cdot)$ is a one-way cryptographic hash function. The verifier with the authentic h_R requests for x_2 and requires the authentication of the received blocks. The prover sends the auxiliary authentication information (AAI) $\Theta_2 = \langle h(x_1), h_b \rangle$ to the verifier. Firstly, The verifier calculates the values of $h(x_2)$, $h_a = h(h(x_1) \parallel h(x_2))$ and $h_R = h(h_a \parallel h_b)$ in order to verify x_2. Then, the verifier calculates h_R, compares it with the authentic h_R, and judges whether they are equal. MHT is not only used to verify the values of data blocks, but also utilized to authenticate the positions of data blocks. We define that the sequence of leaf nodes is the left-to-right. All leaf nodes follow this sequence and are unique in the MHT.

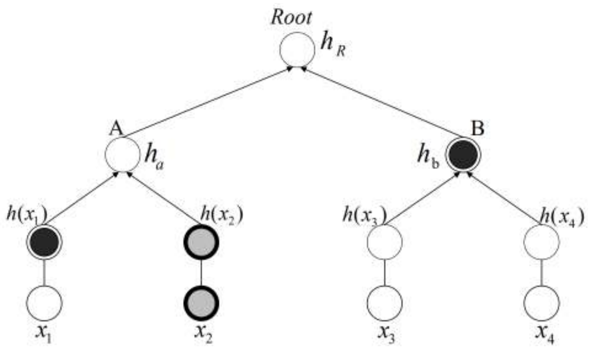


FIGURE 1. Example of the Merkle hash tree authentication of data elements. We treat the leaf nodes $h(x_1), h(x_2), \ldots, h(x_n)$ as the left to right sequence.

IN	*LN*	*Block*
1	[1]	$F^{\varepsilon}_{[1]}$
2	[2]	$F^{\varepsilon}_{[2]}$
3	[3]	$F^{\varepsilon}_{[3]}$
4	[4]	$F^{\varepsilon}_{[4]}$
5	[5]	$F^{\varepsilon}_{[5]}$
6	[6]	*null*

(a)

IN	*LN*	*Block*
1	[1]	$F^{\varepsilon}_{[1]}$
2	[2]	$F^{\varepsilon}_{[2]}$
3	[3]	$F^{\varepsilon}_{[3]}$
4	[6]	F'
5	[5]	$F^{\varepsilon}_{[5]}$
6	[7]	*null*

(b)

IN	*LN*	*Block*
1	[1]	$F^{\varepsilon}_{[1]}$
2	[2]	$F^{\varepsilon}_{[2]}$
3	[3]	$F^{\varepsilon}_{[3]}$
4	[6]	F'
5	[5]	$F^{\varepsilon}_{[5]}$
6	[7]	F''
7	[8]	*null*

(c)

IN	*LN*	*Block*
1	[1]	$F^{\varepsilon}_{[1]}$
2	[2]	$F^{\varepsilon}_{[2]}$
3	[6]	F'
4	[5]	$F^{\varepsilon}_{[5]}$
5	[7]	F''
6	[8]	*null*

(d)

FIGURE 2. (a) Initialization operation of index logic table. (b) Modification operation of index logic table. (c) Insertion operation of index logic table. (d) Deletion operation of index logic table.

D. INDEX LOGIC TABLE (ILT)

In this subsection, we introduce the ILT [15], which is a concrete dynamic structure. There are three columns in the ILT, the 1st and 2nd column refer to index number (*IN*) and logic number (*LN*) of the file blocks, and the values separately are denoted as ξ_i and $[i]$. The 3rd column is the additional column, representing the data blocks, but not appearing in ILT actually, and the value is expressed as $F^{\xi}_{[i]}$.

There are four operations in the ILT, including initialization, modification, insertion, deletion. In the ξ-th ILT, there is a file F^{ξ}, which is split into n blocks, such as $F^{\xi} = (F^{\xi}_{[1]}, F^{\xi}_{[2]}, \ldots, F^{\xi}_{[n]})$, and FIGURE 2 (a) describes an initialization operation (i.e., $n = 6$), FIGURE 2 (b) describes a modification operation, which modifies the data block value into F' at the *IN* value $\phi_1 = 4$, FIGURE 2 (c) describes an insertion operation, which inserts the new data block value F'' after the *IN* value $\phi_2 = 5$, Fig 2 (d) describes a deletion operation, which deletes the data block at the *IN* value $\phi_3 = 3$.

E. FUZZY IDENTITY-BASED SIGNATURE

The fuzzy identity-based signature was proposed in [30]. The scheme represents a user with identity *ID*, who issues the signature results. Subsequently, other users with identity ID' are allowed to perform verification if and only if they are within a certain distance, which is an error tolerance value. The four algorithms included in this scheme are shown as follows:

- *Setup*(1^k): This probabilistic algorithm generates the master key *mk* and the public parameters *pp* by inputting a security parameter 1^k, as well as an error tolerance value d.
- *Extract*(mk, ID): This algorithm generates the private key, which inputs the *mk* and the identity *ID*, correspondingly outputs a secret key sk_{ID} related to *ID*.
- *Sign*(M, sk_{ID}, pp): This probabilistic algorithm generates the signature, which inputs the *pp*, the sk_{ID} related to *ID* and a message M, correspondingly outputs the signature σ.
- *Verify*(σ, M, ID', pp): This deterministic algorithm verifies the correctness of the signature σ. It inputs the public parameters *pp*, the message M, the corresponding signature σ as well as the identity ID' and $|ID' \cap ID| \geq d$. It outputs a bit b, if $b = 1$, means that the signature is valid.

IV. SYSTEM MODEL AND SECURITY MODEL

In this section, the system model, system components and security requirements are provided based on the [14].

A. SYSTEM MODEL

The proposed scheme consists of four entities, namely key generation center (KGC), cloud user, cloud storage server, and third-party auditor (TPA). The KGC outputs the corresponding private key according to the identity set of the users. The TPA performs data integrity auditing according to the requests of the users. The processes of dynamic data auditing are shown as below:

1) The cloud user sends the identity set to the KGC.
2) After verifying the properties of the user's identity set, the KGC issues the corresponding private key to the cloud user.
3) When the users receive the private key from the KGC, then the cloud user preprocesses the data file and stores the metadata into the cloud storage server, and deletes the local data.
4) When the users perform the dynamic operations from two different categories of the data block and ILT, namely modification, insertion and deletion. Then, the users generate the new metadata and send the update requests to the cloud storage server.
5) The TPA interacts with the cloud storage server to audit the data by running a challenge-response protocol.

B. SYSTEM COMPONENTS

Now, we describe eight algorithms in the proposed scheme based on the [14].

- *Setup* $\left(1^k, m, d\right) \rightarrow \left(pp, mk\right)$: The KGC executes this algorithm. It inputs the security parameter k, a maximum value m of describing the dimensions of the identity set, and an error tolerance value d. Then, it outputs the public parameters pp, and the master key mk.
- *Extract* $\left(pp, mk, \omega\right) \rightarrow \widehat{SK}_\omega$: This algorithm is executed by the KGC. It generates a cloud user's secret key $\widehat{SK}_\omega$ by inputting the identity set ω, the public parameters pp and the master key mk.
- *MetadataGen* $\left(pp, \widehat{SK}_\omega, F\right) \rightarrow \left(\Phi, \Gamma, \tau, sig\left(H(R)\right)\right)$: Cloud user runs this algorithm. It inputs the public parameters pp, the private key $\widehat{SK}_\omega$ and a file F. Then, it outputs the data block tag Φ, the index logic table tag Γ, the file tag τ, as well as generating the signature $sig\left(H(R)\right)$ of the root node R.
- *Data block dynamic operation* $\left(F, \widehat{SK}_\omega, pp\right) \rightarrow \left(F', \Phi'\right)$: The users execute this algorithm. It inputs a file F, the private key $\widehat{SK}_\omega$, the public parameters pp. Then it outputs the updated file F' and the updated data block authenticator Φ'.
- *Index logic table dynamic operation* $\left(pp, \widehat{SK}_\omega, Update, F\right) \rightarrow \left(F', \Gamma', sig\left(H(R')\right), Resp_{update}\right)$: This algorithm is executed by the users. It takes a file F, the public parameters pp, the private key $\widehat{SK}_\omega$, and an update request $Update$ as input. It outputs an updated file F', the updated index logic table tag Γ', the updated $sig\left(H(R')\right)$, as well as a response $Resp_{update}$ for the updating operations.
- *Challenge* $\left(pp, \omega'\right) \rightarrow Chal$: The TPA inputs the public parameters pp and the identity set ω' of the users. It outputs the challenge information $Chal$.
- *Response* $\left(pp, F, Chal, \Phi, \Gamma\right) \rightarrow resp$: This algorithm is executed by the cloud storage server. It inputs the public parameters pp, a file F, the challenge information $Chal$, the data block tag Φ and the index logic table tag Γ. It outputs the response information $resp$ to the TPA.
- *Verify* $\left(pp, \omega', Chal, resp\right) \rightarrow TRUE$ or $FALSE$: The TPA runs this algorithm. It inputs the public parameters pp, the identity set ω' of the users, $Chal$ and $resp$. If the file F is unchanged, it outputs the $TRUE$, otherwise outputs the $FALSE$.

C. SECURITY REQUIREMENTS

In the cloud storage system, we presume that KGC is trusted and honest to perform key generation operations. The TPA is semi-trusted, it honestly performs auditing operations, but is curious about the user's information. Cloud storage server in order to maintain their own profits, deliberately conceals the damaged documents from the users or deletes data that is not frequently accessed for the purpose of saving storage space. Therefore, files may be at risk in the cloud storage server, and there are several attacks in the system as follows:

1) Correctness: In the processes of data integrity verification, the TPA first presents the cloud storage server with a challenge, and then the cloud storage server returns challenge-response information. Finally, the TPA verifies the correctness of the response information.
2) Privacy-preserving: In the implementation of the auditing protocol, the TPA receives the response information from the cloud storage server, which is the result of aggregation of data blocks and tags, so that the TPA does not retrieve any specific data block information.
3) Delete-insert attack: We combine the ILT [15] dynamic structure. In the last row, the end flag value does not represent the actual data block. However the value is updated due to the dynamic operations, so that our scheme resists delete-insert attack. Thus the security of the scheme is effectively improved.
4) Forgery attack: In order to maintain its own benefits, the cloud storage server may delete or maliciously tamper with the data blocks. During the verification processes of the auditing, the cloud storage server forges the data blocks in some way, so that the TPA does not detect that the data blocks are corrupted.

V. THE PROPOSED SCHEME

We describe the specific structure of fuzzy identity-based dynamic auditing of big data on cloud storage based on the scheme [14]. Fuzzy identity-based signature has the property of error tolerance, which allows the user issuing a signature with the identity set ω and could be verified with another identity set ω' if and only if they are within a certain distance. Thus, it can provide biometric authentication. Our scheme contains eight algorithms: Setup, Extract, MetadataGen, Data block dynamic operation, Index logic table dynamic operation, Challenge, Response, Verify. The system structure is shown in the FIGURE 3 and the detailed processes are described as below:

A. SETUP

This algorithm is utilized to initialize the system, which is executed by KGC. We assume that G_1 and G_2 are two multiplicative cyclic group with the prime order q. A bilinear pairing map: $e : G_1 \times G_1 \rightarrow G_2$. g is a generator in G_1. $H : \{0, 1\}^* \rightarrow G_1$. The KGC chooses $g_1 = g^\gamma$, $g_2 \in G_1$, $\gamma \in Z_q$ as well as $t_1, t_2, \ldots, t_{m+1} \in G_1$ randomly. We presume that M is the set $\{1, 2, \ldots, m+1\}$, where m is the maximum number of the identity set. We define a function $T(x)$:

$$T(x) = g_2^{x^m} \prod_{i=1}^{m+1} t_i^{\Delta_{i,m}(x)}. \tag{1}$$

Then we select a random $z' \in Z_q$, compute $v' = g^{z'}$. Therefore, the master key is $mk = \gamma$ and the public parameters are obtained as

$$pp = \left\{g_1, g_2, t_1, \ldots, t_{m+1}, v', \Lambda = e(g_1, g_2)\right\}. \tag{2}$$

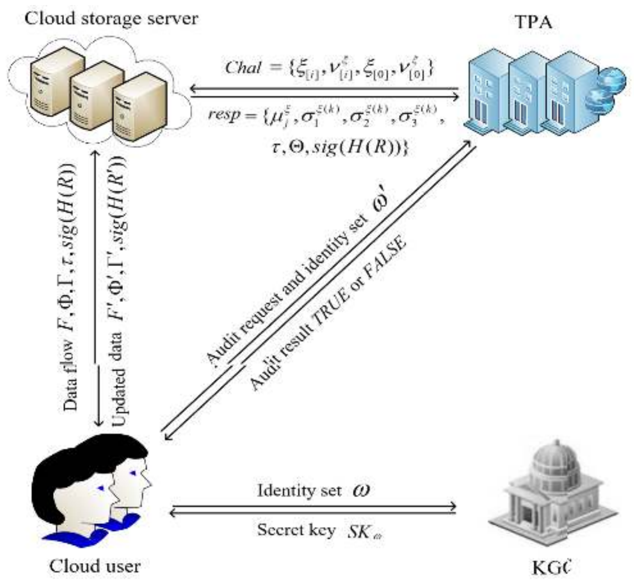

Cloud storage server
TPA
$Chal = \{\xi_{[i]}, v^{\xi}_{[i]}, \xi_{[0]}, v^{\xi}_{[0]}\}$
$resp = \{\mu^{\xi}_{j}, \sigma^{\xi(k)}_{1}, \sigma^{\xi(k)}_{2}, \sigma^{\xi(k)}_{3}, \tau, \Theta, sig(H(R))\}$
Data flow $F, \Phi, \Gamma, \tau, sig(H(R))$
Updated data $F', \Phi', \Gamma', sig(H(R'))$
Audit request and identity set ω'
Audit result TRUE or FALSE
Identity set ω
Secret key SK_{ω}
Cloud user
KGC

FIGURE 3. The system structure of fuzzy identity-based dynamic data integrity auditing protocol.

B. EXTRACT

This algorithm is also executed by KGC in order to generate the private key for the identity set ω, where $|\omega| = m$. Firstly, KGC selects $d-1$ degree polynomial $\boldsymbol{q}$ such that $\boldsymbol{q}(0) = \gamma$, and picks a random number $r_k \in Z_q$ for $(k \in \omega)$. Next, the KGC computes:

$$SK_k = g_2^{\boldsymbol{q}(k)} T(k)^{r_k}, \quad sk_k = g^{-r_k}. \tag{3}$$

Thus, the secret key $\widehat{SK}_{\omega} = (SK_k, sk_k)_{k \in \omega}$, which is corresponding to the identity set ω.

C. METADATAGEN

Give a file F, the users split the file into many blocks, and store them into α ILT separately. Each ILT includes n data blocks, each data block is split s sectors long, i.e., $F = \{f^{\xi}_{[i],j}\} \in Z_q$ for $1 \leqslant \xi \leqslant \alpha, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s$. The users pick $\mathbb{N}$ in Z_q as the file name, s random number $u_1, u_2, \ldots, u_s \in G_1$. We assume that $\widehat{t_0} = \mathbb{N} \parallel u_1 \parallel u_2 \parallel \cdots \parallel u_s$. Then the file tag τ is equal to $\widehat{t_0}$ together with signature [25] on $\widehat{t_0}$: $\tau = \widehat{t_0} \parallel sig(\widehat{t_0})$. For block $[i]_{(1 \leqslant i \leqslant n)}$ in an arbitrary ILT (i.e., ξ-th ILT), the users select a random $s_k \in Z_q$ for $(k \in \omega)$, and input $\widehat{SK}_{\omega} = (SK_k, sk_k)_{k \in \omega}$. Then, the users generate the tag for i-th block in ξ-th ILT as follows:

$$\begin{aligned} \sigma^{\xi\,(k)}_{1,[i]} &= SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel [i]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[i],j}}\right)^{s_k}, \\ \sigma^{\xi\,(k)}_{2,[i]} &= g^{-s_k}, \\ \sigma^{\xi\,(k)}_{3,[i]} &= g^{-r_k}, \end{aligned} \tag{4}$$

for $(k \in \omega)$, and get the corresponding tag of $[i]$-block in ξ-th ILT $\Phi^{\xi}_{[i]} = \left\{\sigma^{\xi\,(k)}_{1,[i]}, \sigma^{\xi\,(k)}_{2,[i]}, \sigma^{\xi\,(k)}_{3,[i]}\right\}$. Next, for each ILT, we utilize the [0]-th block to represent the entire ILT. The users generate the tag for the ξ-th ILT as follows:

$$\sigma^{\xi\,(k)}_{1,[0]} = SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel H(ILT_{\xi}) \parallel [0]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[0],j}}\right)^{s_k} \tag{5}$$

as well as values of $\sigma^{\xi\,(k)}_{2,[0]}$ and $\sigma^{\xi\,(k)}_{3,[0]}$ are the same to $\sigma^{\xi\,(k)}_{2,[i]}$ and $\sigma^{\xi\,(k)}_{3,[i]}$ for $(k \in \omega)$, which are g^{-s_k} and g^{-r_k} respectively. Similarly, the ξ-th ILT tag is $\Gamma^{\xi}_{[0]} = \left\{\sigma^{\xi\,(k)}_{1,[0]}, \sigma^{\xi\,(k)}_{2,[0]}, \sigma^{\xi\,(k)}_{3,[0]}\right\}$. Next, the users compute the $H(ILT_{\xi})$ and calculate a root node R based on the structure of MHT, as well as generating the signature [28] on the root node R: $sig(H(R))$. Finally, the users present the cloud storage server with the file and the corresponding tag $\left\{F, \Phi^{\xi}_{[i]}, \Gamma^{\xi}_{[0]}, \tau, ILT_{\xi}, sig(H(R))\right\}$, then delete the local data.

D. DATA BLOCK DYNAMIC OPERATION

In this subsection, we elaborate on the dynamic operations of this scheme, which contains two categories: the dynamic operations of the data block and the ILT. The dynamic operations consist of the modification ($\boldsymbol{M}$), insertion ($\boldsymbol{I}$) and deletion ($\boldsymbol{D}$). When the data blocks in the ILT are inserted on the upper limit or deleted to be empty, or the data blocks in the ILT are not updated for a long time, users execute dynamic operations of the ILT. We assume that users temporarily store both old and new ILT in local storage, thus, they decide to update the new ILT and root node by themselves, which also reflects the flexibility of the proposed scheme. In the following descriptions, we presume that the file F, data block tag $\Phi^{\xi}_{[i]}$ and ILT tag $\Gamma^{\xi}_{[0]}$ are calculated by users and stored in the cloud storage server as well as root node R is signed and uploaded to the cloud storage server.

1) DATA MODIFICATION

A basic data block modification operation refers to replace one data block with another new. Assume that the users modify the θ_1-th data block to $F^{\xi}_{[\theta_1]'}$ in the ξ-th ILT. Users first split the data block into s sectors, namely $F^{\xi}_{[\theta_1]'} = \{f^{\xi}_{[\theta_1]',1}, f^{\xi}_{[\theta_1]',2}, \ldots, f^{\xi}_{[\theta_1]',s}\}$ on the basis of the new data block, and then generate the new tag:

$$\begin{aligned} \sigma^{\xi\,(k)}_{1,[\theta_1]} &= SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel [\theta_1]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[\theta_1]',j}}\right)^{s_k}, \\ \sigma^{\xi\,(k)}_{2,[\theta_1]} &= g^{-s_k}, \\ \sigma^{\xi\,(k)}_{3,[\theta_1]} &= g^{-r_k}, \end{aligned} \tag{6}$$

for $(k \in \omega)$, and get the corresponding tag $\Phi^{\xi}_{[\theta_1]} = \left\{\sigma^{\xi\,(k)}_{1,[\theta_1]}, \sigma^{\xi\,(k)}_{2,[\theta_1]}, \sigma^{\xi\,(k)}_{3,[\theta_1]}\right\}$. Finally, users update the new data block and tag $\left\{F^{\xi}_{[\theta_1]'}, \Phi^{\xi}_{[\theta_1]}\right\}$.

2) DATA INSERTION

Data block insertion means inserting the new data block after the specific location. Now, we suppose that the users insert the new data block $F^{\xi}_{[\theta_2]''}$ after the (θ_2-1)-th block in the ξ-th ILT. Data block insertion is similar to the modification operation. Firstly, the data block is divided into s sectors long, namely $F^{\xi}_{[\theta_2]''} = \{f^{\xi}_{[\theta_2]'',1}, f^{\xi}_{[\theta_2]'',2}, \ldots, f^{\xi}_{[\theta_2]'',s}\}$, and then the users generate the new tag:

$$\sigma^{\xi\,(k)}_{1,[\theta_2]} = SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel [\theta_2]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[\theta_2]'',j}}\right)^{s_k},$$
$$\sigma^{\xi\,(k)}_{2,[\theta_2]} = g^{-s_k},$$
$$\sigma^{\xi\,(k)}_{3,[\theta_2]} = g^{-r_k}, \qquad (7)$$

for ($k \in \omega$), and get the corresponding tag $\Phi^{\xi}_{[\theta_2]} = \left\{\sigma^{\xi\,(k)}_{1,[\theta_2]}, \sigma^{\xi\,(k)}_{2,[\theta_2]}, \sigma^{\xi\,(k)}_{3,[\theta_2]}\right\}$. Finally, the users update the new data block and tag $\left\{F^{\xi}_{[\theta_2]''}, \Phi^{\xi}_{[\theta_2]}\right\}$.

3) DATA DELETION

The deletion operation is contrary to the insertion operation, it refers to deleting data block at a specific location in an arbitrary ILT. We assume that the users delete the data block at the θ_3-th in the ξ-th ILT. The specific algorithm is similar to modification and insertion operations, so it is omitted.

E. INDEX LOGIC TABLE DYNAMIC OPERATION

When the dynamic operations of the data block are in a stable state, the users perform dynamic operations of the ILT. The details are shown as follows:

1) INDEX LOGIC TABLE MODIFICATION

The dynamic operations of the ILT are similar to the dynamic operations of the data block. We presume that the ξ-th ILT, namely ILT_{ξ} is modified to $ILT_{\xi}{}'$ (i.e., $\xi = 3$, see FIGURE 4). We regard the ILT as [0]-th block. The users first split the [0]-th block into s sectors, and then generate the new tag for $ILT_{\xi}{}'$:

$$\sigma^{\xi'\,(k)}_{1,[0]} = SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel H(ILT_{\xi}{}') \parallel [0]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi'}_{[0],j}}\right)^{s_k},$$
$$\sigma^{\xi'\,(k)}_{2,[0]} = g^{-s_k},$$
$$\sigma^{\xi'\,(k)}_{3,[0]} = g^{-r_k}, \qquad (8)$$

for ($k \in \omega$), and get the corresponding tag $\Gamma^{\xi'}_{[0]} = \left\{\sigma^{\xi'\,(k)}_{1,[0]}, \sigma^{\xi'\,(k)}_{2,[0]}, \sigma^{\xi'\,(k)}_{3,[0]}\right\}$. Then, the users present the cloud storage server with the update request $Update = (\boldsymbol{M}, \xi, ILT_{\xi}{}', \Gamma^{\xi'}_{[0]})$, which $\boldsymbol{M}$ is modification operation. When the cloud storage server receives the update request, it performs $ExecUpdata(F, \Gamma^{\xi}_{[0]}, Updata)$: (1) replaces the ILT_{ξ} to $ILT_{\xi}{}'$, and outputs the new F', (2) replaces the $\Gamma^{\xi}_{[0]}$ to $\Gamma^{\xi'}_{[0]}$, (3) on the basis of the MHT, replaces the $H(ILT_{\xi})$ to $H(ILT_{\xi}{}')$ and calculates the new root node R'.

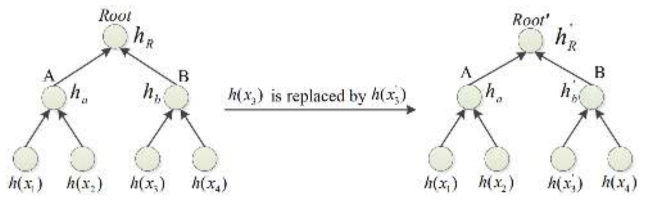


FIGURE 4. Example of modifying the index logic table. Here, x_{ξ} and x'_{ξ} are used to denote $H(ILT_{\xi})$ and $H(ILT'_{\xi})$, respectively.

Finally, the cloud storage server presents the users with a response : $Resp_{updata} = \left(\Theta_{\xi}, sig(H(R)), R'\right)$ in order to prove that the update operations have been completed, which Θ_{ξ} is auxiliary authentication information (AAI) for authentication of ILT_{ξ}. When the users receive the $Resp_{updata}$ from the cloud storage server, they first calculate $H(ILT_{\xi})$ and $H(ILT_{\xi}{}')$ based on the new and old ILT. Next, the users compute the root node R by using $\{\Theta_{\xi}, H(ILT_{\xi})\}$, and determine whether the values of AAI and R are true by verifying the signature $sig(H(R))$. If the verification fails, the output is WRONG, otherwise the users continue to verify if the modification operation is performed as required. The users first calculate the new root node by using $\{\Theta_{\xi}, H(ILT_{\xi}{}')\}$ and compare it with R'. If it outputs TRUE, the users sign the new root node R' by $sig(H(R'))$ and send it to the cloud storage server for updating. Finally, the users perform the auditing protocol. If it outputs TRUE, the users delete the corresponding local storage.

2) INDEX LOGIC TABLE INSERTION

ILT insertion means that a new ILT is inserted after the specific location. We suppose that the users insert the new ILT^{*}_{ξ} after the ξ-th ILT_{ξ} (i.e., $\xi = 3$, see FIGURE 5). Firstly, on the basis of the new ILT^{*}_{ξ}, the users first split it into s sectors, and then generate the new tag:

$$\sigma^{*\,(k)}_{1,[0]} = SK_k \cdot \left(H(\mathbb{N} \parallel \xi^{*} \parallel H(ILT^{*}_{\xi}) \parallel [0]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{*}_{[0],j}}\right)^{s_k},$$
$$\sigma^{*\,(k)}_{2,[0]} = g^{-s_k},$$
$$\sigma^{*\,(k)}_{3,[0]} = g^{-r_k}, \qquad (9)$$

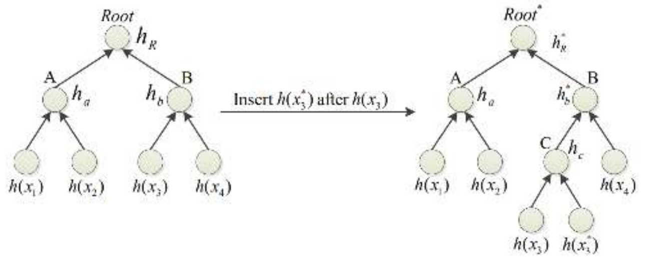


FIGURE 5. Example of inserting the index logic table. Here, x_{ξ} and x^{*}_{ξ} are used to denote $H(ILT_{\xi})$ and $H(ILT^{*}_{\xi})$, respectively.

for ($k \in \omega$), and get the corresponding tag $\Gamma^{*}_{[0]} = \left\{\sigma^{*\,(k)}_{1,[0]}, \sigma^{*\,(k)}_{2,[0]}, \sigma^{*\,(k)}_{3,[0]}\right\}$. Then, the users send the update

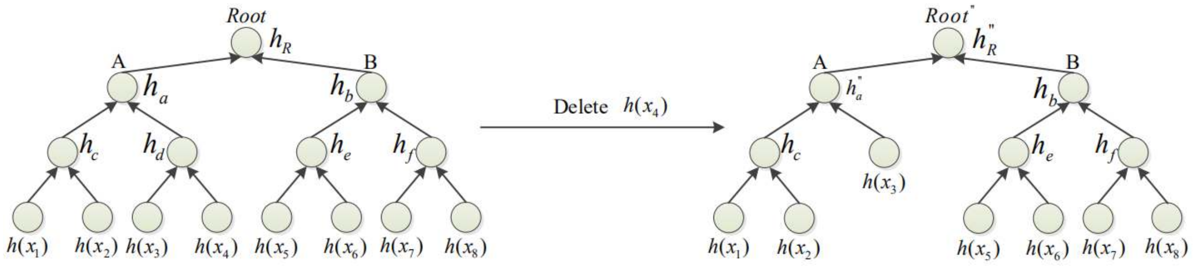


FIGURE 6. Example of deleting the index logic table. Here, x_ξ is used to denote $H(ILT_\xi)$.

request $Update = (\boldsymbol{I}, \xi, ILT_\xi^*, \Gamma_{[0]}^*)$ to the cloud storage server, which $\boldsymbol{I}$ is insertion operation. When the cloud storage server gets the update request, it performs $ExecUpdata(F, \Gamma_{[0]}^\xi, Updata)$: (1) stores ILT_ξ^* and increases a leaf node $h(H(ILT_\xi^*))$ after the leaf node $h(H(ILT_\xi))$ on the MHT as well as updating F^*, (2) increases the $\sigma_{1,[0]}^{*\,(k)}, \sigma_{2,[0]}^{*\,(k)}, \sigma_{3,[0]}^{*\,(k)}$ into the tag set and outputs $\Gamma_{[0]}^*$, (3) on the basis of the structure of the MHT, generates the new root node R^*. Finally, the cloud storage server presents the users with a response : $Resp_{updata} = \left(\Theta_\xi, sig(H(R)), R^*\right)$ to prove that the update operations have been completed, which Θ_ξ is auxiliary authentication information (AAI) for authentication of ILT_ξ. We assume that the value of ξ is 3 (i.e., $\xi = 3$), it means inserting the $h(H(ILT_3^*))$ after the leaf node $h(H(ILT_3))$, and only increasing a leaf node $h(H(ILT_3^*))$ and an internal node c in the original tree structure, which the internal node c is $h_c = h(h(H(ILT_3)) \parallel h(H(ILT_3^*)))$. When the users receive the $Resp_{updata}$ from the cloud storage server, the users first generate $H(ILT_\xi)$ and $H(ILT_\xi^*)$ based on the new and old ILT. Next, the users compute the root node R value by using $\left\{\Theta_\xi, H(ILT_\xi)\right\}$, and determine whether the values of AAI and R are true by verifying the signature $sig(H(R))$. If the verification fails, the output is WRONG, otherwise the users check to see if the insertion operation has been performed as required. The users calculate the new root node by using $\left\{\Theta_\xi, H(ILT_\xi), H(ILT_\xi^*)\right\}$ and compare it with R^*. If it outputs TRUE, the users sign the new root node R^* by $sig(H(R^*))$ and send it to the cloud storage server for updating. Finally, the users perform the auditing protocol. If it outputs TRUE, the users delete the corresponding local storage.

3) INDEX LOGIC TABLE DELETION

The deletion of ILT is contrary to the insertion operation, it refers to deleting the ILT at the specific location (i.e., $\xi = 4$, see FIGURE 6). When the cloud storage server receives the *Update* message, it executes the deletion operation. Firstly, it deletes ILT_ξ and the leaf node $h(H(ILT_\xi))$ on the MHT, then generates the new root node R''. In addition, the specific process of the deletion is similar to operations of modification and insertion, so it is omitted.

F. CHALLENGE

The users with identity set ω' verify the data integrity through sending the form of challenge to the cloud storage server. The TPA first judges whether $\left|\omega \cap \omega'\right| \geqslant d$ holds, if it is true, the TPA forwards the challenge information to the cloud storage server. The processes are provided as follows:

TPA selects the set $L = \{[1], [2], \ldots, [\beta]\}$ randomly, which is the subset of the set $[1, n]$, and picks the subset $I = \{1, 2, \ldots, \upsilon\}$ of the set $[1, \alpha]$. Then, the TPA executes the auditing protocol and wants to challenge an arbitrary block $[i]$ in an arbitrary (i.e., ξ) ILT. For the auditing of the ILT, we use [0]-th block instead. For arbitrary $\xi_{[i]}$ and $\xi_{[0]}$ ($\xi \in I$, $[i] \in L$), TPA randomly selects $\nu_{[i]}^\xi \in Z_q$ and $\nu_{[0]}^\xi \in Z_q$ individually. The *Chal* is the set $\left\{\xi_{[i]}, \nu_{[i]}^\xi, \xi_{[0]}, \nu_{[0]}^\xi\right\}$, where $\xi \in I$, $[i] \in L$. Then the TPA presents the cloud storage server with the *Chal*.

G. RESPONSE

Upon receiving the $Chal = \left\{\xi_{[i]}, \nu_{[i]}^\xi, \xi_{[0]}, \nu_{[0]}^\xi\right\}$ from the TPA, where $\xi \in I$, $[i] \in L$. The cloud storage server calculates the responses are shown as follows:

$$\begin{aligned} \mu_j^\xi &= \sum_{(\xi_{[i]}, \nu_{[i]}^\xi) \in Chal} \nu_{[i]}^\xi \cdot f_{[i],j}^\xi + \nu_{[0]}^\xi \cdot f_{[0],j}^\xi \\ &= \sum_{(\xi_{[\hat{i}]}, \nu_{[\hat{i}]}^\xi) \in Chal} \nu_{[\hat{i}]}^\xi \cdot f_{[\hat{i}],j}^\xi, \end{aligned} \tag{10}$$

for $[\hat{i}] \in \hat{L} = \{[0], [1], [2], \ldots, [\beta]\}$, and corresponding tag:

$$\begin{aligned} \sigma_1^{\xi\,(k)} &= \prod_{(\xi_{[i]}, \nu_{[i]}^\xi) \in Chal} \left(\sigma_{1,[i]}^{\xi\,(k)}\right)^{\nu_{[i]}^\xi} \cdot \left(\sigma_{1,[0]}^{\xi\,(k)}\right)^{\nu_{[0]}^\xi} \\ &= \prod_{(\xi_{[\hat{i}]}, \nu_{[\hat{i}]}^\xi) \in Chal} \left(\sigma_{1,[\hat{i}]}^{\xi\,(k)}\right)^{\nu_{[\hat{i}]}^\xi}, \\ \sigma_2^{\xi\,(k)} &= \sigma_{2,[\hat{i}]}^{\xi\,(k)}, \\ \sigma_3^{\xi\,(k)} &= \sigma_{3,[\hat{i}]}^{\xi\,(k)}, \end{aligned} \tag{11}$$

for ($k \in \omega'$). In addition, the cloud storage server provides some auxiliary information $\left\{\Theta_\xi\right\}_{1 \leqslant \xi \leqslant \alpha}$ to the TPA, which are the node siblings on the path from the leaf nodes $\left\{h(H(ILT_\xi))\right\}_{1 \leqslant \xi \leqslant \alpha}$ to the root node R of the MHT. Finally, the cloud storage server forwards *resp*:

$$\left\{\mu_j^\xi, \sigma_1^{\xi\,(k)}, \sigma_2^{\xi\,(k)}, \sigma_3^{\xi\,(k)}, \tau, \left\{\Theta_\xi\right\}_{1 \leqslant \xi \leqslant \alpha}, sig(H(R))\right\} \tag{12}$$

to the TPA.

TABLE 2. Comparisons summery between existing schemes and our scheme.

Schemes	[14]	[31]	[32]	[33]	Ours
Fuzzy identity-based	√	×	×	×	√
Dynamic structure	×	MVT	×	MHT	MHT-ILT
Resist delete-insert attack	×	×	√	√	√
Synchronization problem	×	×	×	√	×

H. VERIFY

Upon receiving the *resp* from the cloud storage server, the TPA first verifies the τ. If the output is TRUE, the TPA continues to compute the root node R by using $\left\{h(H(ILT_\xi)), \Theta_\xi\right\}_{1\leqslant\xi\leqslant\alpha}$, then validates the signature $sig(H(R))$. If it outputs *FALSE*, the TPA rejects, otherwise, the TPA selects an arbitrary d-element subset S of $\left|\omega\cap\omega'\right|$, and verifies whether formula (13) is equal. If the equation holds, output *TRUE*, otherwise output *FALSE*.

$$\prod_{(\xi_{[\hat{i}]},\nu^{\xi}_{[\hat{i}]})\in Chal} \Lambda^{\nu^{\xi}_{[\hat{i}]}} \overset{?}{=} \prod_{k\in S}\Bigg\{ e(\sigma_1^{\xi\,(k)}, g)\cdot \Big(e(T(k), \{\sigma_3^{\xi\,(k)}\}^{\nu^{\xi}_{[0]}}) \cdot e(H(\mathbb{N}\parallel \xi \parallel H(ILT_\xi)\parallel\ [0])\cdot v')^{\nu^\xi_{[0]}} \cdot \prod_{j=1}^{s} u_j{}^{\mu_j^\xi}, \sigma_2^{\xi\,(k)})\Big) \cdot \prod_{(\xi_{[i]},\nu^\xi_{[i]})\in Chal} \Big(e(T(k), \{\sigma_3^{\xi\,(k)}\}^{\nu^\xi_{[i]}})\cdot e(H(\mathbb{N}\parallel\xi\parallel[i])\cdot v')^{\nu^\xi_{[i]}} \cdot \prod_{j=1}^s u_j{}^{\mu_j^\xi}, \sigma_2^{\xi\,(k)})\Big)\Bigg\}^{\Delta_{k,S}(0)}. \quad (13)$$

VI. SECURITY ANALYSIS

In this section, we verify the correctness of data auditing and present the security analysis of this scheme.

- *Correctness*: The correctness of equation is judged by operation of bilinear pairing. Firstly, we compute the $\sigma_1^{\xi\,(k)}$, which is given as (14). Then, we calculate the value of $\prod_{k\in S} e(\sigma_1^{\xi\,(k)}, g)$, which is given as (15). Therefore, we compute the formula (16), and the equation in verify algorithm holds. We leave the details of formula (14) (15) (16) to Appendix A
- *Privacy-preserving*: In the processes of data integrity auditing, the TPA presents the cloud storage server with the challenge information $Chal = \left\{\xi_{[i]}, \nu^\xi_{[i]}, \xi_{[0]}, \nu^\xi_{[0]}\right\}$ and receives responses information including μ_j^ξ, $\sigma_1^{\xi\,(k)}$, $\sigma_2^{\xi\,(k)}$, $\sigma_3^{\xi\,(k)}$. Even if the TPA collects enough responses information in order to get information about the data blocks. However, the value μ_j^ξ is the result of aggregation of data blocks and random values, and the values $\sigma_1^{\xi\,(k)}$, $\sigma_2^{\xi\,(k)}$, $\sigma_3^{\xi\,(k)}$ are the results of aggregation of tags $\sigma_{1,[i]}^{\xi\,(k)}$, $\sigma_{2,[i]}^{\xi\,(k)}$, $\sigma_{3,[i]}^{\xi\,(k)}$ and random values, respectively. The TPA does not get specific information about data blocks during the processes of auditing. Therefore, the data privacy is preserved and further the security of the auditing is improved.
- *Delete-insert attack*: In our dynamic structure, we introduce the ILT [15], in which the last row is an end flag. Therefore, in FIGURE 2 (c), at the position of $IN = 6$, the corresponding value of LN is the largest, which the value is [7]. If we delete a data block at $IN = 6$ then insert a new data block, the LN value of the new data block becomes [8]. We obtain that there are different LN values when delete and insert data blocks. Thus, our scheme resists delete-insert attack.
- *Forgery attack*: Assume that the cloud storage server forges an illegal data block m^* in the ILT^* instead of the data block m in the ILT, and it is obvious that $m^* \neq m$ and $H(ILT^*) \neq H(ILT)$. When the cloud storage server receives the $Chal$ from the TPA, it provides the responses to the cloud storage server. However, the responses contain the data block m^*, $h(H(ILT^*))$ and auxiliary information Θ^*. Then, the TPA calculates the value of the root node R^* by inputting the $h(H(ILT^*))$ and auxiliary information Θ^*, as well as verifying the signature $sig(H(R))$. If the output is TRUE, it means that the cloud storage server has ability to find a hash function achieving $h(H(ILT^*)) = h(H(ILT))$, but $H(ILT^*) \neq H(ILT)$. According to the collision-resistant capability of hash function, it is a difficult problem and impossible to successfully forge. Therefore, our scheme resists forgery attack.

VII. CHARACTERISTICS AND PERFORMANCE ANALYSIS

In this section, the characteristics comparisons and performance analysis of this article are described.

A. CHARACTERISTICS COMPARISONS

In TABLE 2, we do the comparisons summery between existing schemes [14], [31]–[33] and our scheme. Scheme [31] is a dynamic data integrity auditing based on the MVT, [33] also supports the dynamic operations based on the MHT. However, the scheme of [31] and [33] are not fuzzy identity-based and scheme [33] has the synchronization problem. References [32] and [14] are an identity-based data auditing scheme and a fuzzy identity-based data auditing scheme, respectively. Regrettably, none of them supports dynamic operations. We observe that our scheme is not only achieving a fuzzy identity-based dynamic data integrity verification and supporting the dynamic operations, but also resisting delete-insert attack. Moreover, the MHT combines with time stamp to achieve a dynamic PDP scheme, which leads to a synchronization problem [33]. However, our dynamic structure combines the MHT with the ILT, so that the proposed scheme avoids the synchronization problem.

B. PERFORMANCE ANALYSIS

We achieve our test code on a virtual machine by using the Ubuntu operation system, which is equipped with an Intel Core i5-6600 CPU at 3.30 GHz processor. Assume that $T_{(h)}$ defines the running time of a hash function operation. We run the operation time of the hash function by using the GMP library is about 0.001ms. In addition, figures in this section are drawn by utilizing the MATLAB software.

In the first part, we provide the generation time of the root node hash value during the metadata generation. Assume that file size is 1T, each sector is fixed 160 bits, and each data block is set as 16KB. We set block numbers of each ILT as the abscissa. According to the size of each data block and the block numbers in each ILT, we calculate the total amount of data stored in each ILT, as well as the storage space is occupied by each ILT. Furthermore, the total file size is 1T, so we calculate the number of corresponding ILT, which is equivalent to the number of leaf nodes on the MHT. We presume that the nodes of the MHT are evenly distributed, therefore, the number of all nodes on the MHT can be calculated. In the process of generating metadata, we need to get the hash value of the root node. The number of all hash values has been provided before. One hash operation time $T_{(h)}$ is about 0.001ms, so we calculate the time of returning the hash value of the root node in the metadata generation phase. As can be seen from FIGURE 7, when each ILT is stored 1024 blocks, it only takes about 0.13s.

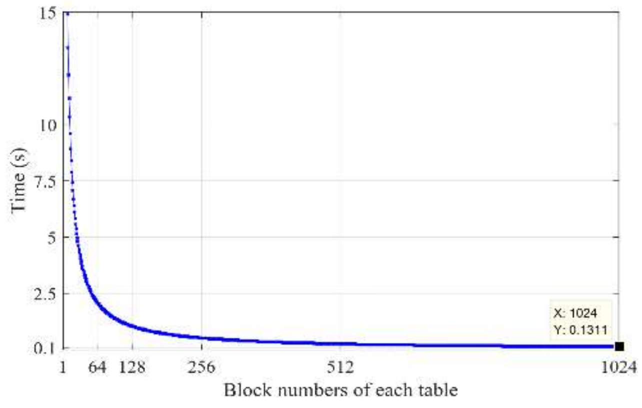


FIGURE 7. Time of generating the root node hash value in the metadata generation process. ***Remark:*** Each ILT corresponds to each leaf node in the MHT. Moreover, the number of data blocks stored in each ILT is the same. For instance, when the abscissa value is set to 512, the time of generating the root node hash value is about 0.26s in the metadata generation process. When the abscissa value is set to 1024, the time of generating the root node hash value is about 0.13s in the metadata generation process. In the following figures, the situations are similar.

We also compare the MHT structure with our scheme, as shown in FIGURE 8. The structure of the MHT is

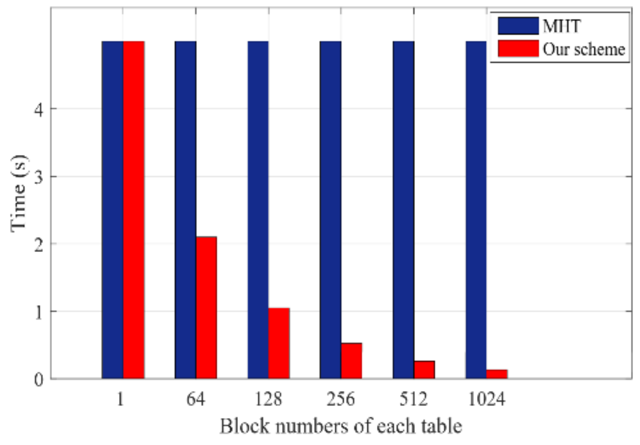


FIGURE 8. Time comparison of generating the root node hash value between the MHT structure and our scheme in the metadata generation process.

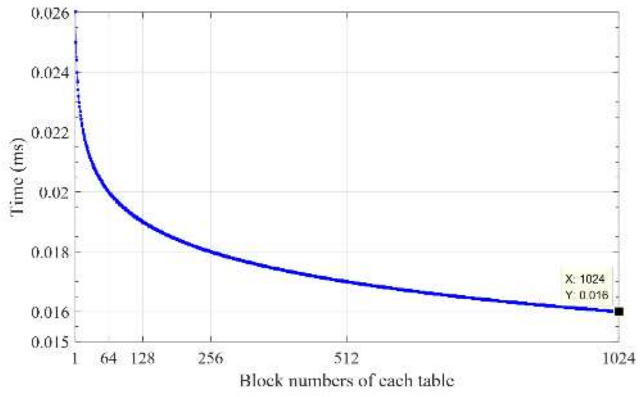


FIGURE 9. Time of updating the root node hash value in the dynamic operations.

equivalent to the case where the abscissa value is 1 in our scheme. For the convenience of comparison, we list bar graphs of the MHT at other positions of the abscissa. Obviously, our scheme not only takes less time than the dynamic structure of the MHT but also is suitable for big data scenarios.

In addition, when the users perform the dynamic operations of the ILT, they update the hash value of the root node. The users receive the corresponding auxiliary authentication information, and then calculate the hash values of path nodes. Therefore, we calculate the corresponding time for updating the hash value of the root node. It is shown in FIGURE 9, we assume that 1024 data blocks are stored in each ILT, updating the root node hash value only takes about 0.016 ms. Similarly, our scheme consumes less time than the dynamic structure of the original MHT.

In the second part, we assume that the users store the new ILT in local storage during the dynamic operations. We analyze the storage cost and communication overhead of the users. The abscissa values are the block numbers of each ILT, so we calculate the total number of the ILT required.

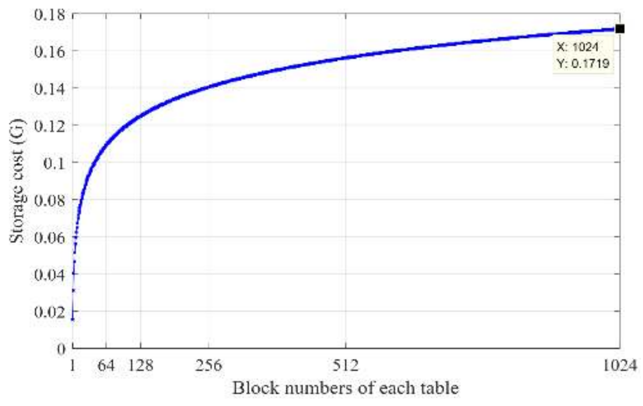


FIGURE 10. Storage cost of the user in the dynamic operations.

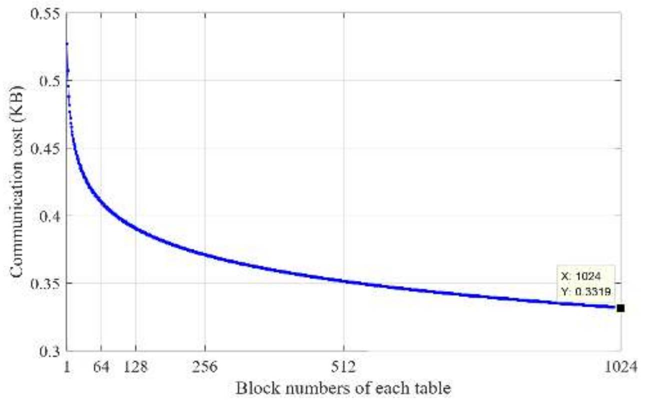


FIGURE 11. Communication cost of the user in the dynamic operations.

Further more, we get the storage cost of each ILT, as well as the storage cost of storing the total ILT. If each ILT stores 1024 data blocks, users probably request about 0.17G storage spaces. It is shown in FIGURE 10.

When the users perform dynamic operations for the ILT, the hash value of the root node is updated. Therefore, the users compute the corresponding hash values of path nodes. Because of the users store the new ILT in local storage, They only need to download the auxiliary authentication information from the cloud storage server, so the communication overhead is greatly reduced. It is shown in FIGURE 11. When 1024 blocks are stored in each table, the communication cost is about 0.33KB for once updating of the root node. Besides, there is only one interaction between the cloud storage server and users.

Moreover, we present a comparison between our scheme and the MHT structure, as shown in FIGURE 12. The structure of the MHT is equivalent to the case where the abscissa value is 1 in our scheme. For the convenience of comparison, we also list the bar graphs of MHT at other positions of the abscissa. It is obvious that our scheme has more advantages in terms of communication cost.

In summary, in FIGURE 7 to FIGURE 12, when the abscissa is set to 1, it means that only one data block is stored in each ILT, which corresponds to the original MHT structure. When the abscissa is set to the right endpoint,

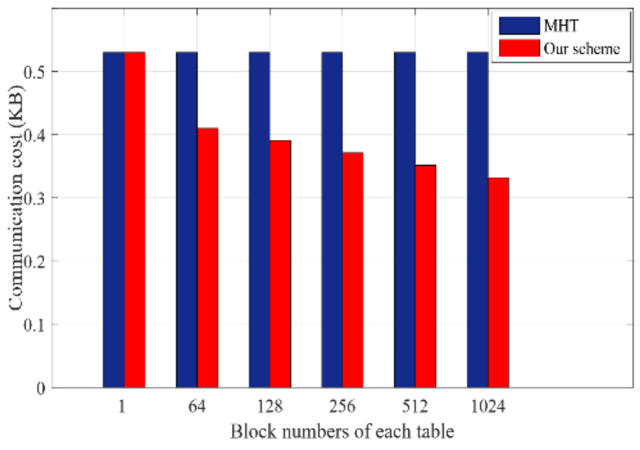


FIGURE 12. Communication cost comparison of the user between the MHT structure and our scheme in the dynamic operations.

it can be approximately regarded as the original ILT structure. Our dynamic structure is a combination of MHT and ILT. From FIGURE 7, FIGURE 8, FIGURE 9, FIGURE 11 and FIGURE 12, the analysis results show that our scheme performances are better than MHT, they are slightly weaker than ILT. However, in FIGURE 10, the storage cost of the proposed scheme is more advantageous than ILT. When the number of blocks stored in ILT increases, computation and communication cost decrease (e.g., FIGURE 7, FIGURE 8, FIGURE 9, FIGURE 11 and FIGURE 12), however, the storage cost increases (e.g., FIGURE 10). On the contrary, when the number of blocks stored in ILT decreases, computation and communication cost increase, but the storage cost decreases. Therefore, according to the conditions of computation, communication and storage cost, the user can flexibly set the number of blocks stored in ILT.

If the users do not store the new ILT in local storage, they only need to save the original hash value of the root node. The storage cost of the users is lower than users storing the new ILT in local storage. Unfortunately, during the dynamic operations, when the users update the hash value of the root node, they need to download the corresponding ILT from the cloud storage server, so that the communication overhead is increased evidently, and the number of interactions between the cloud storage server and users is at least two times, which is also increased.

VIII. CONCLUSION

Cloud storage services are becoming increasingly popular in recent years, therefore, data integrity auditing has also attracted a lot of attentions. In this paper, we proposed a fuzzy identity-based dynamic auditing of big data on cloud storage. The developed dynamic structure is a combination of the MHT and the ILT. On the basis of ensuring the correctness of the data auditing process, the proposed scheme also guarantees that the TPA does not get the specific data information, thus, preserves the privacy of the data. At the same time, the proposed scheme combines the structure of ILT, which effectively resists delete-insert attack. In terms

$$\begin{aligned}\sigma_1^{\xi\,(k)} &= \prod_{(\xi_{[\hat{i}]},\nu^{\xi}_{[\hat{i}]})\in Chal} \left(\sigma^{\xi\,(k)}_{1,[\hat{i}]}\right)^{\nu^{\xi}_{[\hat{i}]}} = \prod_{(\xi_{[i]},\nu^{\xi}_{[i]})} \left\{SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel [i]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[i],j}}\right)^{s_k}\right\}^{\nu^{\xi}_{[i]}} \\ &\quad \cdot \left\{SK_k \cdot \left(H(\mathbb{N} \parallel \xi \parallel H(ILT_\xi) \parallel [0]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[0],j}}\right)^{s_k}\right\}^{\nu^{\xi}_{[0]}} \end{aligned} \tag{14}$$

$$\begin{aligned}\prod_{k\in S} e(\sigma_1^{\xi\,(k)}, g) &= \prod_{k\in S} \prod_{(\xi_{[i]},\nu^{\xi}_{[i]})} e(SK_k, g)^{\nu^{\xi}_{[i]}} \cdot e(H(\mathbb{N} \parallel \xi \parallel [i]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[i],j}}, g^{s_k}) \\ &\quad \cdot e(SK_k, g)^{\nu^{\xi}_{[0]}} \cdot e(H(\mathbb{N} \parallel \xi \parallel H(ILT_\xi) \parallel [0]) \cdot v' \cdot \prod_{j=1}^{s} u_j^{f^{\xi}_{[0],j}}, g^{s_k}). \end{aligned} \tag{15}$$

$$\begin{aligned}&\prod_{k\in S} \left\{e(\sigma_1^{\xi\,(k)}, g) \cdot \prod_{(\xi_{[i]},\nu^{\xi}_{[i]})} e(T_k, \{\sigma_3^{\xi\,(k)}\}^{\nu^{\xi}_{[i]}}) \cdot e\left((H(\mathbb{N} \parallel \xi \parallel [i]) \cdot v')^{\nu^{\xi}_{[i]}} \cdot \prod_{j=1}^{s} u_j{}^{\mu^{\xi}_j}, \sigma_2^{\xi\,(k)})\right)\right. \\ &\quad \left. \cdot e(T_k, \{\sigma_3^{\xi\,(k)}\}^{\nu^{\xi}_{[0]}}) \cdot e\left((H(\mathbb{N} \parallel \xi \parallel H(ILT_\xi) \parallel\ [0]) \cdot v')^{\nu^{\xi}_{[0]}} \cdot \prod_{j=1}^{s} u_j{}^{\mu^{\xi}_j}, \sigma_2^{\xi\,(k)})\right)\right\}^{\Delta_{k,S}(0)} \\ &= \prod_{k\in S} \prod_{(\xi_{[\hat{i}]},\nu^{\xi}_{[\hat{i}]})\in Chal} \left(e(g_2^{\boldsymbol{q}(k)}, g)^{\nu^{\xi}_{[\hat{i}]}}\right)^{\Delta_{k,S}(0)} = \prod_{(\xi_{[\hat{i}]},\nu^{\xi}_{[\hat{i}]})\in Chal} e(g_2, g_1)^{\nu^{\xi}_{[\hat{i}]}} = \prod_{(\xi_{[\hat{i}]},\nu^{\xi}_{[\hat{i}]})\in Chal} \Lambda^{\nu^{\xi}_{[\hat{i}]}}. \end{aligned} \tag{16}$$

of performance analysis, the results showed that our scheme consumed less time than the original MHT, whether it generated the root node during the metadata generation stage or updated the root node during the dynamic operations. We also presented the analysis of users' storage cost and communication overhead in dynamic operations. It can be found that our scheme requires less communication overhead than the MHT structure through comparison. Besides, our scheme needs fewer interactions between users and the cloud storage server.

In the future, we further study the attribute-based cryptosystem and try to construct a new attribute-based dynamic data auditing scheme, combining with more effective dynamic structure.

APPENDIX A SECURITY ANALYSIS

A. CORRECTNESS

See (14)–(16), shown at the top of this page

ACKNOWLEDGMENTS

The authors of this article would like to thank the editor and anonymous readers for their valuable suggestions. This work is supposed by the National Natural Science Foundation of China (Grant NO.61771140, 61972095, U1736112), the Major Science and Technology Project in Fujian Province (Grant NO.2017H6005), the Major Program of Fuzhou Municipal Bureau of Science and Technology (Grant NO.(2017)325), the Enterprise Cooperation Project (Grant NO.DH-1307).

REFERENCES

[1] X. Xu and Q. Hua, "Industrial big data analysis in smart factory: Current status and research strategies," *IEEE Access*, vol. 5, pp. 17543–17551, 2017.

[2] H. Fang, L. Xu, Y. Zou, X. Wang, and K.-K. R. Choo, "Three-stage Stackelberg game for defending against full-duplex active eavesdropping attacks in cooperative communication," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10788–10799, Nov. 2018.

[3] H. Fang, L. Xu, and X. Wang, "Coordinated multiple-relays based physical-layer security improvement: A single-leader multiple-followers stackelberg game scheme," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 197–209, Jan. 2018.

[4] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, to be published, doi: 10.1109/TETC.2019.2904637.

[5] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Inf. Sci.*, vol. 484, pp. 113–134, May 2019.

[6] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2710190.

[7] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Jan. 2016.

[8] C. Guo, S. Su, K.-K. R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2018.2883680.

[9] C. Guo, R. Zhuang, Y. Jie, K.-K. R. Choo, and X. Tang, "Secure range search over encrypted uncertain IoT outsourced data," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1520–1529, Apr. 2019.

[10] C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2768045.

[11] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Oct. 2007, pp. 598–609.

[12] 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.

[13] 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.

[14] 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. Depend. Sec. Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2017.

[15] F. Wang, L. Xu, H. Wang, and Z. Chen, "Identity-based non-repudiable dynamic provable data possession in cloud storage," *Comput. Elect. Eng.*, vol. 69, pp. 521–533, Jul. 2018.

[16] Y. Deswarte, J.-J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Proc. 6th Work. Conf. Integrity Internal Control Inf. Syst. (IICIS)*. Amsterdam, The Netherlands: Springer, Nov. 2004, pp. 1–11.

[17] A. Juels and J. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Nov. 2007, pp. 584–597.

[18] F. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-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.

[19] Y. Zhang and M. Blanton, "Efficient dynamic provable possession of remote data via update trees," *ACM Trans. Storage*, vol. 12, no. 2, Feb. 2016, Art. no. 9.

[20] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2018.2789893.

[21] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.

[22] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2019.2918022.

[23] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2929045.

[24] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious RAM," in *Proc. EUROCRYPT*, Mar. 2013, pp. 279–295.

[25] B. Shao, G. Bian, Y. Wang, S. Su, and C. Guo, "Dynamic data integrity auditing method supporting privacy protection in vehicular cloud environment," *IEEE Access*, vol. 6, pp. 43785–43797, 2018.

[26] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vols. 343–344, pp. 1–14, May 2016.

[27] Y. Yu, Y. Zhang, Y. Mu, W. Susilo, and H. Liu, "Provably secure identity based provable data possession," in *Provable Security*. 2015, pp. 310–325.

[28] A. Ara, M. Al-Rodhaan, Y. Tian, and A. Al-Dhelaan, "A secure privacy-preserving data aggregation scheme based on bilinear ElGamal cryptosystem for remote health monitoring systems," *IEEE Access*, vol. 5, pp. 12601–12617, 2017.

[29] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[30] P. Yang, Z. Cao, and X. Dong, "Fuzzy identity based signature with applications to biometric authentication," *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 532–540, Jul. 2011.

[31] 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, Mar. 2015.

[32] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar. 2015.

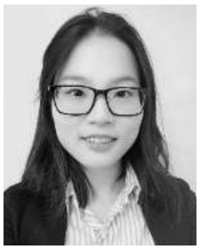
[33] Z. Mo, Y. Zhou, S. Chen, and C. Xu, "Enabling non-repudiable data possession verification in cloud storage systems," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jul. 2014, pp. 232–239.

CHENBIN ZHAO received the B.S. degree in information and computing science from Quanzhou Normal University, Fujian, China, in 2017. He is currently pursuing the M.S. degree with the College of Mathematics and Informatics, Fujian Normal University, Fujian. His research interests include cryptography and information security.

LI XU received the B.S. and M.S. degrees in mathematics from Fujian Normal University, in 1992 and 2001, respectively, and the Ph.D. degree in information and communication engineering from the Nanjing University of Posts and Telecommunications, in 2004. He is currently a Professor and the Ph.D. Supervisor with the School of Mathematics and Informatics, Fujian Normal University, where he is also the Director of the Information Construction and Management Office and the Key Laboratory of Network Security and Cryptography in Fujian Province. He has published over 150 articles in refereed journals and conferences. His research interests include wireless networks and communication, network optimization and information security, complex networks and systems, and intelligent information in communication networks. He is also a member of ACM and a Senior Member of CCF and CIE, China. He has been invited to act as the PC Chair or a member in more than 30 international conferences.

JIGUO LI received the B.S. degree in mathematics from Heilongjiang University, Harbin, China, in 1996, and the M.S. degree in mathematics and the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, in 2000 and 2003, respectively. From September 2006 to March 2007, he was a Visiting Scholar with the Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia. From February 2013 to January 2014, he was a Visiting Scholar with the Institute for Cyber Security, The University of Texas at San Antonio. He is currently a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, and the College of Computer and Information, Hohai University, Nanjing, China. His research interests include cryptography and information security, cloud computing, wireless security, and trusted computing. He has published over 150 research articles in refereed international conferences and journals. His work has been cited more than 3000 times at Google Scholar. He has served as a Program Committee Member in over 20 international conferences and served as a Reviewer for over 90 international journals and conferences.

FENG WANG received the M.S. degree in applied mathematics from Guangzhou University, in 2006. He is currently pursuing the Ph.D. degree in applied mathematics with Fujian Normal University. He is also an Associate Professor with the College of Mathematics and Physics, Fujian University of Technology. His research interests include computer cryptography, networks, and information security.



HE FANG received the B.Sc. and Ph.D. degrees in applied mathematics from Fujian Normal University, China, in 2012 and 2018, respectively. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Western University, Canada.

She is also involved in distributed security management in IoT and blockchain systems under practical network constraints. Her research interests include intelligent security provisioning, machine learning, as well as distributed optimization and collaboration techniques, and development of new machine-learning enabled authentication schemes through utilization of time-varying wireless environment for security enhancement.

• • •