IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 20, 2020, accepted July 31, 2020, date of publication August 5, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014510

Multi-Replica and Multi-Cloud Data Public Audit Scheme Based on Blockchain

XIAODONG YANG[1], (Member, IEEE), XIZHEN PEI[1], MEIDING WANG[1], TING LI[1], AND CAIFEN WANG[1,2]

[1]College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China
[2]College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China

Corresponding author: Xiaodong Yang (y200888@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61662069 and Grant 61562077; in part by the China Postdoctoral Science Foundation under Grant 2017M610817; in part by the Science and Technology Project of Lanzhou City of China under Grant 2013-4-22; and in part by the Foundation for Excellent Young Teachers by Northwest Normal University under Grant NWNU-LKQN-14-7.

ABSTRACT Cloud storage services provide convenient data storage services for individuals and enterprises. Data owners can remotely access and update outsourcing data. But there are still many security problems, such as data integrity. Although the public audit schemes allow users to authorize third-party auditors (TPA) to verify the integrity of cloud data, there are still a series of problems in the existing public audit schemes. First of all, most of the existing schemes are based on the traditional or identity public key infrastructure. There is a problem of certificate management or key escrow. And they do not support dynamic data update and user identity tracking for group users. Then, existing multi-replica data public audit schemes store all replicas on a cloud storage server. Once the cloud server fails, all replicas will be damaged. Finally, most existing schemes require TPA to be trusted. In practice, TPA may deviate from the public audit protocol or collude with cloud servers to deceive users. To solve these problems, we propose a certificateless multi-replica and multi-cloud data public audit scheme based on blockchain technology. In our scheme, the dynamic hash table and modification record table are introduced to achieve dynamic update of group user data and identity tracking. All replicas are stored in different cloud servers, and their integrity can be audited at the same time. In addition, we use the unpredictability of blocks in the blockchain to construct fair challenge information, thereby preventing malicious TPA and cloud servers from colluding to deceive users. Each audit result is written into the blockchain, which is convenient for users to audit the behavior of TPA. The analysis results show that our proposed scheme is secure in the random oracle model and has higher efficiency in communication and computation cost compared with similar schemes.

INDEX TERMS Blockchain, certificateless cryptosystem, cloud storage, data dynamic update, identity tracking, multi-cloud, multi-replica.

I. INTRODUCTION

In recent years, cloud computing as a new computing model has attracted people's extensive attention [1]. More and more enterprises, such as Alibaba, Amazon, Microsoft, IBM etc. have established their own cloud computing services and opened them to the world. These services provide users with an efficient and flexible data management method, and reduce the burden of local data storage and maintenance [2]. It is a general tendency to store data on the cloud servers.

However, once the data is uploaded to the cloud server, the data owner loses the physical control over the data [3]. Although the cloud server promises that the data will be well maintained. Due to the complexity of the cloud environment, the cloud server is vulnerable to attack from external adversaries and internal hardware or software failures, which may lead to data destruction or even loss [4]. In addition, the cloud server may violate service level agreements and delete data that some users rarely access for economic benefits. Therefore, users must regularly verify the integrity of outsourced data [5], [6].

Public audit technology enables users to outsource data integrity verification to specialized TPA. The TPA regularly

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Afendee Mohamed.

verify data integrity and notify users. Once the verification fails, the data may be corrupted [7]. However, the most existing public audit schemes are based on traditional public key infrastructure (PKI) technology, and facing certificate storage, distribution, revocation and verification. This leads to a lot of cost [8]. To avoid these problems, an identity based cryptosystem (IBC) is proposed [9]. Its security is depended on the trust of the private key generator. There is a key escrow problem, which makes it not suitable for large scale network environment. Certificateless cryptography [10] simultaneously avoids the problems of certificate management and key escrow compared with PKI and IBC. In a certificateless scheme, the user's private key is composed of a partial key generated by key generation center (KGC) and a secret value generated by user. KGC cannot get the full private key of users, which solves the key escrow problem.

Furthermore, with the increasing popularity of team collaboration in cloud computing, group user sharing data audit has become a new hot topic in the field of cloud audit [11], [12]. In practical applications, a few members may modify some shared data maliciously for their own benefit, which will destroy the availability of shared data to a large extent. However, shared data audit puts forward higher requirements for integrity audit. Such as user revocation, data dynamic update and user identity tracking. It means the data owner can track all modification operations and reveal the identity of the user who misbehaves when necessary. Unfortunately, some existing shared data auditing schemes do not consider the issue of identity tracking and the dynamic update of data requires a lot of overhead. Therefore, it is significant to design a multi-user public audit scheme based on certificateless.

To improve the durability and availability of data, users usually create multiple replicas. If one replica is damaged, users can recover data from other replicas. Unfortunately, some existing schemes [13]–[15], only support to verify the integrity of single data. For multiple replicas, you must run r times to verify the integrity of r replicas [16]–[18]. To solve this problem, a series of multi-replica public audit schemes [19]–[21] are proposed. But they store all replicas on a cloud storage server, and once the server fails, they still face the risk of data loss. Therefore, it is significant to design a multi-user, multi-replica and multi-cloud public audit scheme based on certificateless.

In addition, TPA is considered honest and reliable in most public audit schemes. This is a very powerful assumption. In practice, in order to reduce the overhead of verification, an irresponsible TPA may generate a good integrity report without performing any public audit or collude with the cloud server to verify only the complete data blocks to cheat users. In addition, malicious TPA may delay regular audits due to system errors and network failures, causing complete loss of data. Therefore, it is necessary to formulate a public audit scheme to restrict the behavior of auditors. The public audit technology based on blockchain can effectively audit the behavior of TPA. TPA generates challenge information based on unpredictability nonce of the block, and writes the verification result of each time into the blockchain as a transaction.Then users regularly audit TPA behavior based on the data on blockchain. This can not only resist the collusion attack of TPA and server, but also ensure that TPA aduits the integrity of cloud data in a prescribed time.

A. OUR CONTRIBUTIONS

In this paper, we design a new scheme based on blockchain technology. Specifically, our contributions are as follows.

- We design a multi-replica and multi-cloud public audit scheme based on certificateless cryptosystem, which not only avoids the problem of certificate management in PKI, but also solves the key escrow problem in IBC. At the same time, different replicas are stored on different cloud servers, and the TPA can simultaneously audit all replicas on different cloud servers.
- Our scheme supports dynamic data updating and user identity tracking. In our scheme, the improved dynamic hash table is used to achieve the dynamic update of multi replica data, At the same time, the data modification record table is used to realize the traceability of group user identity.
- Our scheme can resist malicious auditor. In a period of time, TPA utilizes the unpredictability of nonce in each block in the blockchain to generate fair challenge information and interacts with the cloud server to verify the data integrity, and sends each audit result to the blockchain as a transaction. Users utilize the immutability, traceability and time sensitivity of the data in the blockchain to verify whether the TPA performed the cloud data integrity audit correctly and on time in a longer period.
- We prove the security of the scheme based on the CDH and DL assumptions under the random oracle model. Including the unforgeability of the signature and the robustness of the audit. Meanwhile, we prove that our scheme can resist malicious auditors. Furthermore, we conduct a comprehensive performance analysis, and the experimental results show that the scheme has a good efficiency in communication and computation overhead.

B. ORGANIZATION

The remainder of this paper is organized as follows. Section II introduces related work. Section III reviews some preliminaries. Sections IV and V respectively present the system model and detailed description of our scheme. Section VI analyzes the security and performance of our scheme. Section VII proposes conclusion.

II. RELATED WORK

In order to ensure the integrity of outsourced data, Juels *et al*. [22] first presented the ''Proofs Of Retrievability'' (POR) mechanism, but this scheme does not consider public audit, and data owner must bear a heavy audit burden. To support public audit, Ateniese *et al*. [23] presented the ''Provable Data Possession'' (PDP) mechanism based on

RSA signature, which introduced an independent TPA to verify the integrity of outsourced data on behalf of users, greatly reducing unnecessary overhead. Since then, many public audit schemes based on homomorphic signature technology [24], [25] have been proposed successively. However, in many practical applications, data owners want specific users to check files in cloud storage. In view of this, a PDP protocol with designated verifier was proposed by Ren *et al*. [26], but this scheme cannot resist replay attack. In order to overcome this shortcoming, a new designated verifier audit scheme was constructed by Yan *et al*. [27].

However, the above schemes cannot support the dynamic update of data. To support the dynamic update of data, Erway *et al*. [28] presented a full dynamic data integrity audit scheme by introduced rank-based authentication skip list. Wang *et al*. [29] proposed a data integrity verification scheme by introduced merkle hash tree (MHT). Zhu *et al*. [30] constructed another public auditing scheme based on index hash table (IHT). However, the above three schemes generate a lot of computation and communication cost during the update and verification process. To solve this problem, a new structure called dynamic hash table (DHT) was proposed by Tian *et al*. [31], and used this structure update the cloud data with high efficiency. However, the scheme does not consider the confidentiality of data. Therefore, Hwang *et al*. [32] constructed a public audit scheme that supports data confidentiality and data dynamic operations.

However, the above schemes are all single user data integrity verification. In order to support group users sharing data audit, Wang *et al*. [11] based on group signature technology constructed a group shared data public audit scheme, but there is a problem of low efficiency. Therefore, Wang *et al*. [12] further adopted ring signature technology to propose a scheme for cloud shared data that supports privacy protection. Unfortunately, neither of the above two schemes consider the issue of group user revocation. In view of this, Wang *et al*. [33] and Luo *et al*. [34] presented cloud shared data public audit schemes supporting user revocation based on proxy resignature technology and secret sharing technology respectively. The confidentiality of data is not considered in the above schemes. Li *et al*. proposed two different authentication schemes [35], [36], and ensured the confidentiality of data. However, the above schemes all adopt certificate based cryptosystem, so there is a problem of certificate management. To avoids this problem, Li *et al*. [37] constructed an identity based privacy protection public auditing scheme. But, this scheme supports single users. In view of this, Yu *et al*. [38] proposed an identity based public audit scheme for shared data supporting privacy protection, but the scheme does not support dynamic update of group data. Yuan *et al*. [39] designed an identity-based group data sharing audit scheme that supports dynamic update of data. However, the above two schemes exist the trouble of key escrow, which has great limitations in practical applications. Therefore, a certificateless public audit scheme for group users shared data was presented by Li *et al*. [40].

Furthermore, key exposure is an important security issue in cloud audit. In recent years, a scheme with key-exposure resistance was designed by Yu *et al*. [41]. This scheme uses a key update technology based on binary tree structure to protect the security of the authenticator generated that earlier than the key exposure time period. However, the security of the authenticator generated after the key exposure time cannot be maintained. In view of this, Yu *et al*. [42] further constructed a strong key-exposure resilient auditing scheme. In this scheme the key exposure in one period does not affect the security of cloud storage auditing in other periods.

The above schemes do not have backup storage for important data, it will cause huge economic losses once lost. To improve the reliability of data storage, the most widespread method is to store multiple replicas of data in the cloud. Curtmola *et al*. [16] and Li *et al*. [18] respectively proposed a multi-replica integrity audit scheme (MR-PDP). However, these two schemes need to check the replicas one by one, which require a lot of computation overhead. To improve efficiency, a new privacy protected MR-PDP scheme was constructed by Hao *et al*. [19], but this scheme does not consider dynamic update of data. In order to achieve dynamic update of multi-replica data, Zhang *et al*. [20] and Peng *et al*. [21] designed a MR-PDP scheme based on rank MHT and position merkle tree (PMT) respectively. However, in the above MR-PDP schemes, all replicas are stored on a cloud server, when a server fails, data will still be damaged or even lost. In order to further improve the security of multi-replica data, Li *et al*. [43] presented a new multi-replica public audit scheme, which stores the replica data in different cloud servers. But this scheme does not support the dynamic update of data. Therefore, it is very significant to design a multi-replica and multi-cloud data public audit scheme based on certificateless cryptosystem for group users.

In addition, for the existing scheme, TPA is considered fully credible. This is a very bold assumption. In practice, a malicious TPA may reduce the number of audits to reduce resource consumption, or collude with cloud server to obtain some benefits. In view of this, Armknecht *et al*. [44] and Xue *et al*. [45] respectively designed a public integrity verification scheme against malicious auditors. However, none of the above schemes can resist the delayed auditors. To solve this problem, Zhang *et al*. [46] presented a new certificateless public auditing scheme based on blockchain technology, which can resist malicious and prolonged auditors, but it requires large computation overhead and does not support dynamic data update.

III. PRELIMINARIES

In this section, we review the notations and definitions related to the proposed scheme.

A. BILINEAR MAPS

Let G_1 and G_2 are two multiplicative cyclic groups of prime order p, and g is a generator of G_1. The $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it satisfies the following properties.

1) Bilinear: For any $a, b \in Z_p$, there is $e(g^a, g^b) = e(g, g)^{ab}$.
2) Non-degenerate: $e(g, g) \neq 1$.
3) Computable: For any $g_1, g_2 \in G_1$, there is an efficient algorithm to calculate $e(g_1, g_2)$.

B. COMPLEXITY ASSUMPTION

Computational Diffie-Hellman (CDH) problem: Given $(g, g^a, g^b) \in G_1$, calculate $g^{ab} \in G_1$.

Definition 1 (CDH Assumption): If any polynomial time algorithm cannot solve the CDH problem on G_1, then this problem on G_1 is difficult.

Discrete Logarithm (DL) Problem: Given $(g, g^a) \in G_1$, calculate $a \in Z_p^*$.

Definition 2 (DL Assumption): If any polynomial time algorithm cannot solve the DL problem on G_1, then this problem on G_1 is difficult.

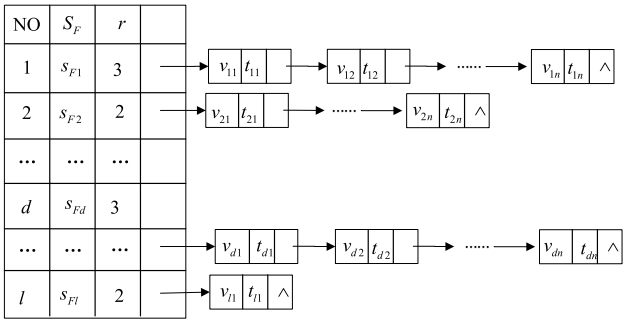

FIGURE 1. Improved dynamic hash table (IDHT).

C. IMPROVED DYNAMIC HASH TABLE

As shown in Figure 1, in a dynamic hash table, there are two types of elements, called file element and data block element [31], [47]. The file element include file index NO, file identifier S_F, and a pointer to the first data block of the file. In order to make it suitable for multi-replica files, we increase the number of replicas r at the file element. Each file is stored in a chain, and the file element serves as the head node of the linked list. The data block element includes the current version of the block v_{li}, a timestamp t_{li}, and a pointer to the next block. v_{li} defaults to 1, when the data block is in the initial state.

There are two types of operations on DHT, block operations and file operations, which include search, modify, insert and delete. These specific process is similar to the linked list operation. Specifically, searching for a block refers to sequentially retrieving the accessed data blocks from the first data block. Modifying the data block need update the corresponding block element directly. Inserting a block after an existing block requires retrieving the given node and inserting a new node after it, and updating the pointer. Deleting a block requires retrieving a given node and delete it. Searching for a file is based on its file identifier to locate file elements, while other file operations will involve operations on file elements and block elements. Specifically, modifying file requires updating file elements and related block elements. Inserting file refers to inserting the linked list composed of file elements and corresponding block elements. Deleting files requires deleting both file elements and data block elements.

D. MODIFICATION RECORD TABLE

As shown in Figure 2, it is a two-dimensional data structure [47], which records the related operations of all data blocks in a file since the latest successful verification. Each modified data block contains a data block identifier and two pointers, which respectively point to the next modified data block and the first modification operation of the current data block. Each operation for the same data block is linked into a linked list in reverse chronological order. Each operation block contains three elements and a pointer, the first element is the user identity, the second element is the relevant operation op (such as: modify, insert, delete) on the data block, and the third element is the operation time t when the operation is performed. The pointer points to the last operation of the data block.

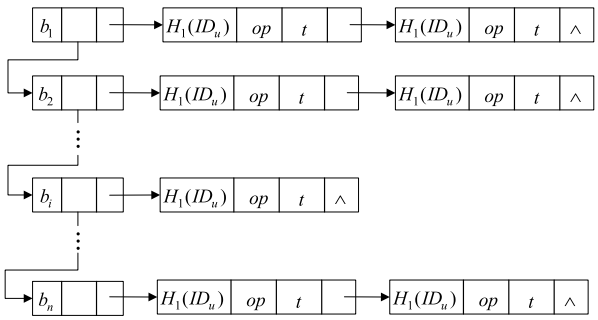

FIGURE 2. Modification record table (MRT).

E. BLOCKCHAIN

Blockchain is famous for its outstanding performance in various cryptocurrency systems (such as bitcoin [48] and Ethereum [49]). Blockchain is a linear collection of data elements, where each data element is called a block. All blocks are linked in chronological order to form a chain, and the encryption hash function is used for security protection. As shown in Figure 3, each block contains the hash value of the current block (BlockHash), the previous block hash value (PreBlockHash), a random number (Nonce), the time stamp of the current block added to the blockchain (Time), the root node value of the Merkel hash tree (MerkleRoot), and multiple transaction records (Tx).

In the blockchain, the participants who verify the validity of a transaction are called miners. Before generating

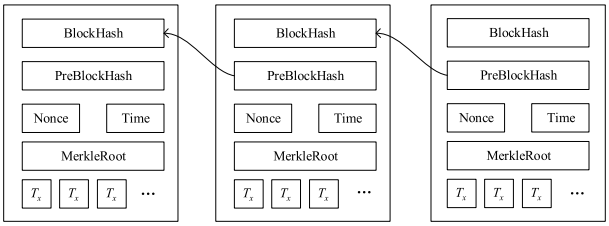

FIGURE 3. Blockchain.

new blocks, miners will collect as many transactions as possible, and find solutions to a difficult problem until get effective nonce. This process is the Proof-of-Work (PoW), also known as ''mining''. The first miner who finds nonce broadcasts the transaction with this nonce in the blockchain system. Other miners verify whether nonce is an effective solution to this difficult problem, thereby adding new block to their blockchain. A transaction can be recorded in the blockchain if it is verified and accepted by a considerable number of miners. For other technical details of blockchain, see [50], [51]. Therefore, blockchain has inherent verifiability, and the transactions recorded in blockchain are non-tamperable.

In order to avoid the loss caused by the 51 percent attack, it is generally believed that transactions confirmed before the latest six blocks ($\varphi = 6$) are basically unalterable. The complete randomness of the nonce in the blockchain guarantees the unpredictability of the block. In our scheme, the nonce of the last φ blocks in the current blockchain is used to generate a unforgeable and unpredictable challenge message. And after the audit, the TPA broadcasts a log file containing the audit results to the network and adds it to the blockchain. The tamperability and openness of the blockchain makes the data stored in the blockchain unable to be destroyed by malicious CSP or TPA and has traceability.

If the block containing the transaction Tx is accepted by most miners and linked to the blockchain, the string Tx will be given a timestamp. This means that Tx is generated no later than when the block is linked to the blockchain. In addition, the average time of block mining in the blockchain is determined, and it is generally considered that a block is generated every 10 minutes. Therefore, transactions in the blockchain are time sensitive. This feature of blockchain can ensure that TPA verifies the integrity of cloud data according to the specified time, and discovers the damage of cloud data as early as possible.

IV. SYSTEM MODEL AND SECURITY MODEL

In this section, we propose the system model and security model of our scheme.

A. SYSTEM MODEL

There are five entities in the system model of our scheme: users (data owner), key generation center (KGC), cloud server providers (CSP), third party auditor (TPA) and blockchain system, as illustrated in Figure 4.

- **User.** The user is responsible for generating data tags, transmitting data to the cloud service provider and dynamically updating the cloud data. Meanwhile, authorize TPA to periodically verify the integrity of cloud data.
- **Third Party Auditor (TPA).** It responsible for verifing the integrity of cloud data, and writing verification results to log files and broadcasting to the blockchain.
- **Cloud service provider (CSP).** It responsible for providing cloud storage services to users and responding to TPA authentication requests. It not only has a large storage space, but also has a huge computing power. In this paper, CSP is composed of cloud server organizer (CO) and cloud servers (CS). The CO is responsible for transferring replicas to the CS, and sending the challenge information to the CS when receiving the challenge information sends by the TPA. After obtaining the evidence returned by the CS, the CO aggregates the data and sends it to the TPA. The CS is responsible for storing data.
- **Key generation center (KGC).** It responsible for generating part of the private key for the user and sending it to the user through a secure channel.
- **Blockchain system.** It responsible for helping TPA generates unpredictable challenge information and records the audit results of TPA. In addition, it also helps users verify the behavior of TPA.

Here, the relationship between the entities in the system model is briefly introduced. After the user uploads the local data to CO, CO will send different replicas of the user to different CS for storage. In addition, users can access and update outsourcing data in anytime. In order to ensure the integrity of the data, the users delegate the TPA to periodically audit the data and verify the audit results of TPA for a long time.

Definition 3: Our scheme consists of five algorithms, Setup, Partial Key Generation, Secret Value Generation, Data Upload, Audit.

Setup: This algorithm is performed by the KGC to generate the master key and public parameters used in the following algorithm.

Partial Key Generation: KGC performs this algorithm to obtain the partial key for users. It inputs the master key and the identity of the user, outputs the partial key.

Secret Value Generation: This algorithm is executed by the user to obtain the secret value and public key. The algorithm randomly selects S_u as the secret value and calculates public key pk_u for the user.

Data Upload: This algorithm enables a user to outsource the data to CSP. The user generates replica files and calculates tags for all data blocks and sends it to the cloud server. Of course, the cloud server should also ensure that the uploaded data is correct.

Audit: This algorithm requires TPA to regularly audit the integrity of cloud data, and users to verify the behavior of TPA in a longer period. TPA sends challenge information to

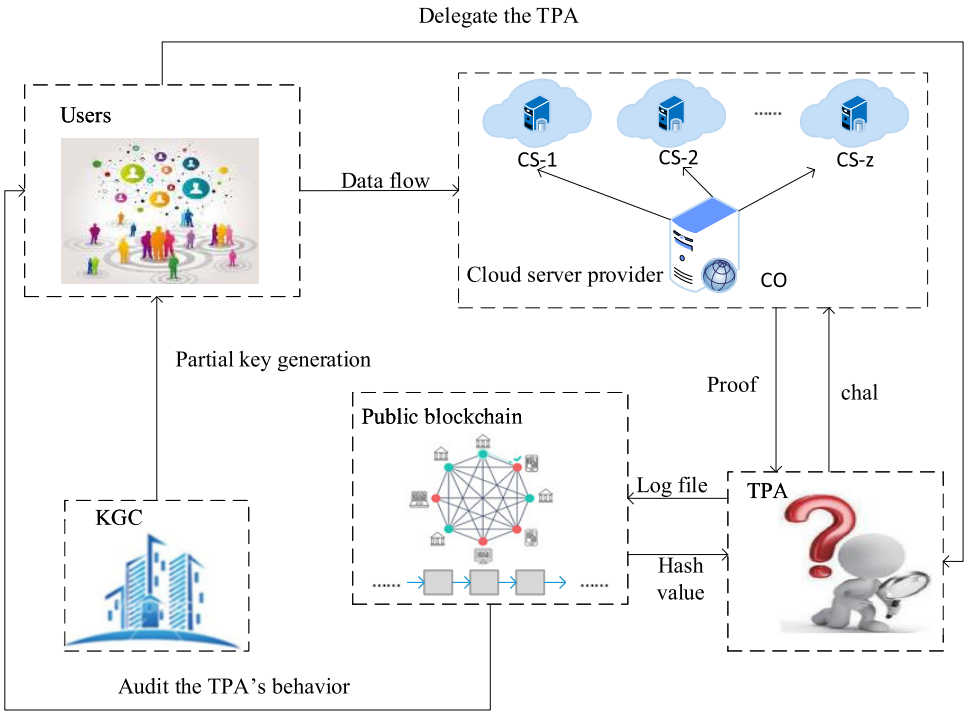

FIGURE 4. System model.

CSP, CSP generates data integrity proof, and TPA verifies the correctness of the proof. Furthermore, This algorithm enables TPA to generate a log file, which records the verification information of TPA, and allows the user to audit the behavior of TPA by checking the validity and correctness of the log file.

B. SECURITY MODEL

We will consider three types of adversaries, called adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$. The $\mathcal{A}_1$ and $\mathcal{A}_2$ are two kinds of adversaries of certificateless cryptosystem [10]. Their goal is to forge the tag of data block. The goal of $\mathcal{A}_3$ is to forge proof. The specific description is as follows.

- The first type of adversary $\mathcal{A}_1$, he can replace the public key of user, but cannot access the master key.
- The second type of adversary $\mathcal{A}_2$, he can access the master key, but cannot perform public key replacement attack.
- The third type of adversary $\mathcal{A}_3$, he can forge data integrity proof to deceive TPA.

We prove the safety of the scheme through the following three games, which involve Challenger B and adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ respectively.

Game 1: $\mathcal{B}$ and $\mathcal{A}_1$ play this game.

Setup: $\mathcal{B}$ runs the Setup algorithm to generate master key and public parameters. $\mathcal{B}$ secretly saves the master key and sends public parameters to $\mathcal{A}_1$.

RepGen: $\mathcal{B}$ runs the RepGen algorithm, gets all replicas of the original file, and returns them to $\mathcal{A}_1$.

Queries: $\mathcal{A}_1$ can perform the following queries to $\mathcal{B}$.

1) Hash-query: For identity ID, $\mathcal{A}_1$ adaptively executes hash-queries to $\mathcal{B}$. $\mathcal{B}$ responses the hash values to $\mathcal{A}_1$.
2) PartialKey-Query: For identity ID, $\mathcal{A}_1$ adaptively executes PartialKey-Query to $\mathcal{B}$. $\mathcal{B}$ generates the partial key by running the Partial Key Generation algorithm and returns it to $\mathcal{A}_1$.
3) SecretValue-Query: For identity ID, $\mathcal{A}_1$ adaptively executes SecretValue-Query to $\mathcal{B}$. $\mathcal{B}$ generates the secret value by running the Secret Value Generation algorithm and returns it to $\mathcal{A}_1$.
4) PublicKey-Query: For any identity ID, $\mathcal{A}_1$ adaptively executes PublicKey-Query to $\mathcal{B}$. $\mathcal{B}$ calculates the public key of the identity ID and returns it to $\mathcal{A}_1$.
5) PublicKey-Replacement: For any identity ID, $\mathcal{A}_1$ can replace its public key by randomly choosing a value.
6) Tag-Query: $\mathcal{A}_1$ adaptively selects any block of any replica to query the corresponding tag under any identity ID. $\mathcal{B}$ obtains the tag by running the TagGen algorithm and returns it to $\mathcal{A}_1$.

Forge: $\mathcal{A}_1$ forges a tag σ'_{ij} for the data block b_{ij} with the identity ID' and the public key $pk_{ID'}$. $\mathcal{A}_1$ wins the game, if the following conditions are satisfied.

1) The tag is valid for identity ID' and public key $pk_{ID'}$.
2) For identity ID', $\mathcal{A}_1$ does not execute PartialKey-Query.
3) For identity ID', $\mathcal{A}_1$ does not execute SecretValue-Query and PublicKey-Replacement.
4) For identity ID' and data block b_{ij}, $\mathcal{A}_1$ does not execute Tag-Query.

Game 2: $\mathcal{B}$ and $\mathcal{A}_2$ play this game.

Setup: $\mathcal{B}$ generates master key and public parameters by running the Setup algorithm and sends them to $\mathcal{A}_2$.

RepGen: $\mathcal{B}$ runs the RepGen algorithm, gets all replicas of the original file, and returns them to $\mathcal{A}_2$.

Queries: $\mathcal{A}_2$ can perform the following queries to $\mathcal{B}$.

1) Hash-query: For identity ID, $\mathcal{A}_2$ adaptively executes hash-queries to $\mathcal{B}$. $\mathcal{B}$ responses the hash values to $\mathcal{A}_2$.
2) PartialKey-Query: For identity ID, $\mathcal{A}_2$ adaptively executes PartialKey-Query to $\mathcal{B}$. $\mathcal{B}$ runs the Partial Key generation algorithm to generate partial key and sends it to $\mathcal{A}_2$.
3) SecretValue-Query: For identity ID, $\mathcal{A}_2$ adaptively executes SecretValue-Query to $\mathcal{B}$. $\mathcal{B}$ generates secret value by running the Secret Value Generation algorithm, and sends it to $\mathcal{A}_2$.
4) PublicKey-Query: For any identity ID, $\mathcal{A}_2$ adaptively executes PublicKey-Query to $\mathcal{B}$. $\mathcal{B}$ calculates the public key of the identity ID and returns it to $\mathcal{A}_2$.
5) Tag-Query: $\mathcal{A}_2$ adaptively selects any block of any replica to query the corresponding tag under any identity ID. $\mathcal{B}$ obtains the tag by running the TagGen algorithm and returns it to $\mathcal{A}_2$.

Forge: $\mathcal{A}_2$ forges a tag σ'_{ij} for the data block b_{ij} with the identity ID'. $\mathcal{A}_2$ wins the game, if the following conditions are satisfied.

1) The tag is valid for identity ID',.
2) For identity ID', $\mathcal{A}_2$ does not execute SecretValue-Query.
3) For identity ID' and data block b_{ij}, $\mathcal{A}_2$ does not execute Tag-Query.

Game 3: $\mathcal{B}$ and $\mathcal{A}_3$ play this game. $\mathcal{A}_3$ stands for dishonest CSP, which may hide data corruption events from users. This game is to prove whether CSP can forge proof to pass the verification. There is only one difference between this game and Game 1. Challenger $\mathcal{B}$ generates challenge information to $\mathcal{A}_3$. $\mathcal{A}_3$ forges an evidence to challenger $\mathcal{B}$.

Challenge: $\mathcal{B}$ generates random challenge information chal and sends it to $\mathcal{A}_3$, requesting $\mathcal{A}_3$ to return a proof of data integrity.

Forge: $\mathcal{A}_3$ forges a proof based on incomplete or erroneous data, and sends it to $\mathcal{B}$. $\mathcal{A}_3$ wins the game, if the forged proof can pass the integrity verification.

V. A NEW PUBLIC AUDIT SCHEME FOR CLOUD DATA

A. DESCRIPTION OF THE SCHEME

In this section, we propose a new public audit scheme for cloud data based on blockchain technology.

1) **Setup:** Enter security parameters, the KGC selects two cyclic groups G_1 and G_2 of prime p, a generator g of G_1 and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. Next, it chooses two pseudo-random functions f_1 and f_2, one pseudo-random permutation π, five anti-collision hash functions $H_1 : \{0,1\}^l \rightarrow G_1, H_2 : \{0,1\}^* \rightarrow G_1, H_3(\cdot)$, $H_4(\cdot)$ and $H_5 : \{0,1\}^* \rightarrow Z_p^*$, where $H_3(\cdot)$ maps $\{0,1\}^*$ to the key space of $\pi_{key}(\cdot)$ and $H_4(\cdot)$ maps $\{0,1\}^*$ to the key space of $f_{2key}(\cdot)$. Then, it randomly selects $\alpha \in Z_p^*$ as the master private key, and calculates $mpk = g^\alpha$ as the main public key. Finally, it keeps the master private key in secret and exposes the system parameters $params = \{G_1, G_2, p, g, e, f_1, f_2, \pi, H_1, H_2, H_3, H_4, H_5, mpk\}$.
2) **Partial Key Generation:** In order to generate partial private key of users, KGC calculates $D_u = H_1(ID_u)^\alpha$ and sends D_u to the user through a secret channel.
3) **Secret Value Generation:** The user randomly selects $S_u \in Z_p^*$ as the secret value and calculates $pk_u = g^{S_u}$ as the public key. Get user's full private key as $sku = (S_u, D_u)$.
4) **Data Upload:** In order to upload the data to the cloud servers, the user performs the following phases.
   - phase 1 (Replica Generation): The user divides the file F into blocks and obtains $m_i \in Z_p (1 \le i \le n)$. Then he selects a random value $\tau_i \in Z_p^*$ and calculates $b_{ij} = m_i + f_{1\tau_i}(i||j), j = 0, \cdots, r$, where f_1 is a pseudo-random function. Finally, each data block b_{ij} is further divided into sectors to obtain $b_{ijk}, 1 \le k \le s$.
   - phase 2 (Tag generation): In order to generate file identification and data tags. The user performs the following steps.
     (a) For each F, the user generates a file identification $S_F = IDS(name \parallel pk_u)$.
     (b) Calculates $\sigma_{ij} = D_u^{\sum_{k=1}^{s} b_{ijk}} H_2(w_{ij})^{S_u}$, where $w_{ij} = (v_i||t_i||i||j)$. The data tag set is $\phi = \{\sigma_{ij}\}_{1 \le i \le n, 1 \le j \le c}$.
     (c) Sends $\{S_F, k, \{\phi\}, \{b_{ij}\}\}$ to CO and delete the local data file.
   - phase 3 (Data upload): After receiving the data uploaded by the user, the CO checks whether the following equation $e(\sigma_{ij}, g) = e(H_1(ID_u)^{\sum_{k=1}^{s} b_{ijk}}, mpk) \cdot e(H_2(w_{ij}), pk_u)$ hold. If the equation holds, the tag is valid. The CO sends file replicas and corresponding tags to different CS, and records the storage location of the replica in the replica record table, as shown in Table 1. The CS saves the data uploaded by the CO.

TABLE 1. Replica record table.

S_F	CS_{id}
S_{F1}	$CS_{id1}, CS_{id2}, CS_{id3}$
S_{F2}	CS_{id1}, CS_{id2}
$\cdots$	$\cdots$
S_{Fl}	$CS_{id1}, CS_{id2}, CS_{id4}$

5) **Audit:** In order to verify the integrity of the outsourced data, TPA and CSP perform the following phases.
   - phase 1 (Challenge): Based on the current time, TPA extracts $\{nc_{l-\varphi+1}, nc_{l-\varphi+2}, \cdots, nc_l\}$ from the blockchain. Then it sends $chal = \{\{nc_{l-\varphi+1}, nc_{l-\varphi+2}, \cdots, nc_l\}, l\}$ to CO, where l indicates the depth of the current blockchain.

- phase 2 (Proof Generation): The CO checks if $\{nc_{l-\varphi+1}, nc_{l-\varphi+2}, \cdots, nc_l\}$ is valid in the blockchain. If it is invalid, the CO rejects it. Otherwise, performs the following steps.
  (a) The CO calculates $\kappa_1 = H_3(nc_{l-\varphi+1} \parallel nc_{l-\varphi+2} \parallel \cdots \parallel nc_l)$, $\kappa_2 = H_4(nc_{l-\varphi+1} \parallel nc_{l-\varphi+2} \parallel \cdots \parallel nc_l)$, $i_\varsigma = \pi_{\kappa_1}(\varsigma)$, $v_{i\varsigma} = f_{2\kappa_2}(\varsigma)$ $\varsigma = 1, \cdots, c$. CO queries the replica record table to find the CS stored in the verified file and sends $(i_\varsigma, v_{i\varsigma})$ to the corresponding CS.
  (b) After each CS receives the $(i_\varsigma, v_{i\varsigma})$ from CO, it calculates $\sigma_{C_j} = \prod_{\varsigma=1}^{c} \sigma_{i\varsigma}^{v_{i\varsigma}}$ and $\varphi_{C_j} = \sum_{k=1}^{s} \sum_{\varsigma=1}^{c} v_{i\varsigma} b_{i\varsigma k}$, and sends $(\sigma_{C_j}, \varphi_{C_j})$ to CO.
  (c) The CO calculates $\sigma = \prod_{j=1}^{r} \sigma_{C_i}$ and $\varphi = \sum_{j=1}^{r} \varphi_{C_i}$, sends $proof = (\sigma, \varphi)$ to TPA.
- phase 3 (Proof Verify): After TPA receives the *proof* sent by CO, The CO performs the following steps.
  (a) Verifies whether the following equation hold:

$$e(\sigma, g) = e((H_1(ID_u)^{\varphi}, mpk) \; e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i \parallel t_i \parallel i_\varsigma \parallel j)^{v_{i\varsigma}}, pk_u). \tag{1}$$

  The data is complete, if the equation is hold.
  (b) For the verification results, TPA generates a log entity $\{nc_{l-\varphi+1}, nc_{l-\varphi+2}, \cdots, nc_l, t, \sigma, \varphi, 1/0\}$ and stores the log entity in the log file. As illustrated in Table 2.

TABLE 2. Log file.

nonces	time	proof	audit result
$\{nc_{l-\varphi+1}^{(1)}, nc_{l-\varphi+2}^{(1)}, \cdots, nc_l^{(1)}\}$	$t^{(1)}$	$(\sigma^{(1)}, \varphi^{(1)})$	1/0
$\{nc_{l-\varphi+1}^{(2)}, nc_{l-\varphi+2}^{(2)}, \cdots, nc_l^{(2)}\}$	$t^{(2)}$	$\sigma^{(2)}, \varphi^{(2)})$	1/0
...	...	...	...
$\{nc_{l-\varphi+1}^{(m)}, nc_{l-\varphi+2}^{(m)}, \cdots, nc_l^{(m)}\}$	$t^{(m)}$	$\sigma^{(m)}, \varphi^{(m)})$	1/0

  (c) Creates a transaction T_x. Its data field is *th*, where $th = H_5(nc_{l-\varphi+1}, nc_{l-\varphi+2}, \cdots, nc_l, t, \sigma, \varphi, 1/0)$. And uploads it to the blockchain. As shown in Figure 5.
- phase 4 (Log Verify): In order to verify the validity of the log files, the user performs the following steps.
  (a) Randomly selects d entities to form a challenge set $C = \{c_1, c_2, \cdots c_d\}$ and verify the accuracy of each entity's time in turn. The user retrieve the block with random value of nc_l, and the block that records the corresponding transaction of the entity on the blockchain. The user extracts the time of these two blocks, and get the approximate time of TPA audit. Then verify whether the TPA audits the integrity of the cloud data according to the agreed time. If the time matches, continue to the next step, otherwise return false.
  (b) Verify whether the equation is valid:

$$e(\prod_{\theta=1}^{d} \sigma^{C_\theta}, g) = e((\prod_{\theta=1}^{d} H_1(ID_u)^{\varphi^{C_\theta}}, mpk) \; e(\prod_{\theta=1}^{d}\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i||t_i||i_\varsigma||j))^{v_{i\varsigma}{}^{C_\theta}}, pk_u). \tag{2}$$

  If the equation is valid, TPA correctly performs the verification of cloud data. Inform TPA to delete log files saved during this period. Otherwise, it returns false.

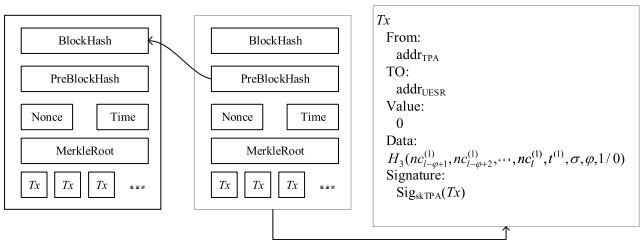

FIGURE 5. Write blockchain.

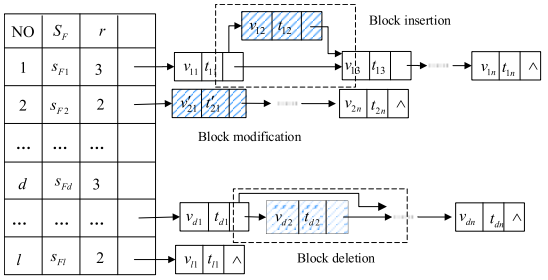

FIGURE 6. Data dynamic operation.

B. DATA DYNAMIC UPDATE

In this paper, data modification includes file modification and block modification. Here we only describe the modification of the block. The modification, insertion and deletion of the block are shown in Figure 6. The specific process is as follows.

1) Block modification: Assume that the ith data block m_i of file F is modified to m_i'. The user runs the replica generation algorithm to generate replicas b_{ij}' and calculates the tag σ_{ij}'. The user generates data information (v_i', t_i') and sends update request (S_F, M, i, v_i', t_i') to TPA and $(S_F, M, i, b_{ij}', \sigma_{ij}')$ to CO. The TPA queries the records in DHT, finds the ith node in linked list of the

file F, and changes it to (v_i', t_i'). The CO queries the replica record table, finds the servers where the modified file is stored, and sends the modification information $(S_F, M, i, b_{ij}', \sigma_{ij}')$ to the corresponding CS. When CS receives the modification information, it replaces b_{ij} with b_{ij}' and updates σ_{ij} to σ_{ij}'. The CO records the operation in the modification record table.

2) Block insertion: Assume that the user need to insert data block after block m_i of file F. As with the modification operation, the user generates replicas of the data block, the corresponding tag and data information. Sends insert request (S_F, I, i, v_i', t_i') to TPA $(S_F, I, i, b_{ij}', \sigma_{ij}')$ to CO. TPA inserts a new node after the ith node of the file F linked list. CO sends the insertion information to the corresponding CS, and CS inserts the corresponding data block after receiving the request. CO records the operation in the modification record table.
3) Block deletion: Assume that the ith data block m_i of file F needs to be deleted. The user sends a deletion information (S_F, D, i) to TPA and CSP. TPA finds and deletes the ith node of the file F linked list in DHT. In addition, CO sends deletion information to the corresponding CS. After receiving the request, the CS deletes the corresponding data block. The CO records the operation in a modification record table.

C. IDENTIFY TRACKING

We use the method of [36] to implement user identity tracking. The only difference is that the CO keeps the modification record table. This avoids the problem of excessive group administrator privileges. In fact, CO maintains a MRT for each file. When a user wants to modify a data block of file F, he needs to send a request to CO. After receiving the request, CO performs the following operations.

- Queries the MRT of file F. If it does not exist, CO creates a new MRT for the file. Otherwise, proceed to the next step.
- Queries whether the modified block exists in the MRT. If it exists, CO inserts a new operation record on the block operation of the block. It includes the identity of the block, the operation performed and the time when the operation was performed. Otherwise, CO inserts a block identifier b_i and an operation record on its block operation.

CO can find dishonest members by looking up the MRT of the file, when there is an argument about the operation of the file F.

D. CORRECTNESS

The proof generated by CSP based on correct data can pass the TPA audit. The correctness of our scheme is based on equation (1) and (2). Since the verification of these two equations is essentially the same, the correctness of equation (1) is shown as follows.

$$\begin{aligned} & e(\sigma, g) \\ &= e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c}\sigma_{i\varsigma}{}^{v_{i\varsigma}}, g) \\ &= e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c}(D_u{}^{\sum_{k=1}^{s} b_{ijk}} \cdot H_2(w_{i\varsigma})^{S_u})^{v_{i\varsigma}}, g) \\ &= e((H_1(ID_u)^{\sum_{j=1}^{r}\sum_{\varsigma=1}^{c}\sum_{k=1}^{s} b_{ijk} v_{i\varsigma}}, g^{\alpha}) \\ &\quad e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i||t_i||i\varsigma||j)^{v_{i\varsigma}}, g^{S_u}) \\ &= e((H_1(ID_u)^{\varphi}, mpk) e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i||t_i||i\varsigma||j)^{v_{i\varsigma}}, pk_u). \end{aligned} \quad (3)$$

VI. SECURITY AND PERFORMANCE ANALYSIS

A. SECURITY ANALYSIS

Theorem 1: Under the adaptive selection message attack, the tag is existentially unforgeable under the random oracle model.

Proof: In order to prove the above theorem, we play two kinds of games, aiming at two types of adversaries of certificateless cryptosystem.

Game 1: If there is a PPT adversary $\mathcal{A}_1$ who makes H1-Query, PartialKey-Query, SecretValue-Query, PublicKey-Query, PublicKey-Replace, H2-Query, Tag-Query at most times respectively and successful forged signature. Then there is a challenger $\mathcal{B}$ who can solve the CDH problem with a non-negligible probability ε within time t. Given an example of a CDH problem (g, g^a, g^b), $\mathcal{A}_1$ and $\mathcal{B}$ perform the following security game, and calculate g^{ab}.

Setup: $\mathcal{B}$ generates system parameters by running the Setup algorithm, sets $mpk = g^a$, returns system parameters and mpk to $\mathcal{A}_1$.

RepGen: $\mathcal{B}$ runs the RepGen algorithm, gets all replicas of the original file, and returns them to $\mathcal{A}_1$.

H1-query: For identity ID, $\mathcal{A}_1$ adaptive execution H1-Query. $\mathcal{B}$ saves a list $L_1 \rightarrow \{(ID, h_1, Q, \eta)\}$, if ID in the L_1 list, $\mathcal{B}$ extracts the corresponding list (ID, h_1, Q, η) and sends it to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ chooses $h_1 \in Z_p^*$ and throws a random coin $\eta \in \{0, 1\}$, assume the probability of $\eta = 0$ is ω, then the probability of $\eta = 1$ is $1 - \omega$. If $\eta = 0$, calculates $Q = g^{h_1}$. If $\eta = 1$, calculates $Q = (g^b)^{h_1}$, returns Q to $\mathcal{A}_1$ and inserts (ID, h_1, Q, η) into L_1.

PartialKey-Query: For identity ID, $\mathcal{A}_1$ adaptive execution PartialKey-Query. $\mathcal{B}$ saves a list $L_2 \rightarrow \{(ID, D_{ID}, pk_{ID}, S_{ID})\}$. $\mathcal{B}$ queries the list L_1, and if (ID, h_1, Q, η) does not exist in the list L_1, $\mathcal{B}$ executes H1-Query. When the corresponding value of (ID, h_1, Q, η) is obtained, $\mathcal{B}$ queries the value of η, if $\eta = 1$, $\mathcal{B}$ terminates, otherwise $\mathcal{B}$ executes the following operation.

- If ID exists in the list L_2 and $D_{ID} \neq \perp$, $\mathcal{B}$ will extract D_{ID} and return it to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ extracts (ID, h_1, Q, η)

from the list L_1, calculates $D_{ID} = Q^a = (g^{h_1})^a = (g^a)^{h_1}$ to $\mathcal{A}1$, and written D_{ID} to the corresponding tuple.

- If the ID does not exist in the list L_2, $\mathcal{B}$ extracts (ID, h_1, Q, η) from the list L_1, computes $D_{ID} = Q^a = (g^{h_1})^a = (g^a)^{h_1}$ to $\mathcal{A}_1$, and inserts the new tuple $(ID, D_{ID}, \perp, \perp)$ into the list L_2.

SecretValue-Query: For identity ID, $\mathcal{A}_1$ adaptive execution Secretvalue-Query. $\mathcal{B}$ queries the list L_1, and if (ID, h_1, Q, η) does not exist in the list L_1, $\mathcal{B}$ executes H1-Query, then queries the list L_2.

- If ID exists in the list L_2 and $S_{ID} \neq \perp$, $\mathcal{B}$ directly extracts S_{ID} and return it to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly selects $x \in Z_p^*$, and sets $S_{ID} = x, pk_{ID} = g^x$. Then $\mathcal{B}$ writes S_{ID}, pk_{ID} to the corresponding tuple, and return S_{ID} to $\mathcal{A}_1$.
- If ID does not exist in the list, $\mathcal{B}$ randomly selects $x \in Z_p^*$, sets $S_{ID} = x$, $pk_{ID} = g^x$, and inserts the new tuple $(ID, \perp, S_{ID}, pk_{ID})$ into the list L_2, and returns S_{ID} to $\mathcal{A}_1$.

PublicKey-Query: For identity ID, $\mathcal{A}_1$ adaptive execution Publickey-Query.

- If ID exists in the list L_2 and $pk_{ID} \neq \perp$, $\mathcal{B}$ directly extract pk_{ID} and return it to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ selects $x \in Z_p^*$ randomly, sets $S_{ID} = x$, $pk_{ID} = g^x$, then returns pk_{ID} to $\mathcal{A}_1$, and writes S_{ID}, pk_{ID} to the corresponding tuple.
- If ID does not exist in the list L_2, $\mathcal{B}$ randomly selects $x \in Z_p^*$, sets $S_{ID} = x$, $pk_{ID} = g^x$, and inserts the new tuple $(ID, \perp, S_{ID}, pk_{ID})$ into the list L_2, and returns pk_{ID} to $\mathcal{A}_1$.

Publickey-replace: For (ID, pk_{ID}), $\mathcal{A}_1$ adaptive execution PublicKey-replace.

- If ID exists in the list L_2, $\mathcal{B}$ updates $(ID, D_{ID}, pk'_{ID}, \perp)$.
- If ID does not exist in the list L_2, $\mathcal{B}$ adds a new tuple $(ID, \perp, \perp, pk_{ID})$ to the list L_2.

H2-Query: $\mathcal{A}_1$ adaptive execution H2-Query for w. $\mathcal{B}$ saves a list $L_3 \rightarrow \{(w, h_2, y)\}$. If w exists in the list L_3, $\mathcal{B}$ extracts the corresponding y to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly selects $h_2 \in Z_p^*$, evaluates $y = g^{h_2^*}$ to $\mathcal{A}_1$ and inserted (w, h_2, y) into the list L_3.

Tag-Query: For identity ID, $\mathcal{A}_1$ adaptive execution Tag-Query. $\mathcal{A}_1$ sents (w, b_{ij}, ID) to $\mathcal{B}$, $\mathcal{B}$ queries the list $L_1 \rightarrow \{(ID, h_1, Q, \eta)\}$ and $L_2 \rightarrow \{(ID, D_{ID}, pk_{ID}, S_{ID})\}$, if the corresponding values do not exist, $\mathcal{B}$ executes H1-Query and H2-Query to get the corresponding values. If $\eta = 1$, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ extracts the D_{ID} and S_{ID} from the list L_2, y from the list L_3, and calculates the corresponding tag $\sigma_{ij} = ((g^a)^{h_1})^{\sum_{k=1}^{s} b_{ijk}} \cdot y^{S_{ID}}$ to $\mathcal{A}_1$.

Forge: $\mathcal{A}_1$ forge a tag based on identity ID' and its data block b_{ij}. The requested block b_{ij} has not performed a Tag-Query.

Analysis: If $\mathcal{A}_1$ win the game successfully, $\mathcal{B}$ can get $e(\sigma'_{ij}, g) = e(H_1(ID')^{\sum_{k=1}^{s} b_{ijk}}, mpk) \cdot e(H_2(w'), pk_{ID'})$, further get $e(\sigma'_{ij}, g) = e(g^{bh'_1 \sum_{k=1}^{s} b_{ijk}}, g^a) \cdot e(g^{h'_2}, pk_{ID'})$, so we can get $g^{ab} = \left(\frac{\sigma'_{ij}}{(pk_{ID'})^{h'_2}}\right)^{1/h'_1 \sum_{k=1}^{s} b_{ijk}}$. Next we analyze the probability that $\mathcal{B}$ not interrupt. From the above analysis, we can get the interrupt only happens in PartialKey-Query and Tag-Query, and the probability of $\mathcal{B}$ not interrupte is $(1-\omega)^{q_p+q_T}$, so the probability of $\mathcal{A}_1$ win the game in time $t' \leq t + O(q_{h_1} + q_p + q_s + q_{pk} + q_{pr} + q_{h_2} + q_T)$ is $\varepsilon' \geq \varepsilon \cdot \omega \cdot (1-\omega)^{q_p+q_T} \geq \varepsilon / ((q_p + q_T) \cdot 2e)$.

Game 2: If there is a PPT adversary $\mathcal{A}_2$ who makes H1-Query, SecretValue-Query, PublicKey-Query, H2-Query and Tag-Query at most times respectively and successful forged signature. Then there is a challenger $\mathcal{B}$ who can solve the CDH problem with a non-negligible probability ε within time t. Given an example of a CDH problem (g, g^a, g^b), $\mathcal{B}$ and $\mathcal{A}_2$ perform the following security game, and calculate g^{ab}.

Setup: $\mathcal{B}$ randomly chooses $\chi \in Z_p^*$ as the system master key and sends the system master key and public parameters to $\mathcal{A}_2$.

RepGen: $\mathcal{B}$ runs the RepGen algorithm, gets all replicas of the original file, and returns them to $\mathcal{A}_2$.

H1-query: For identity ID, $\mathcal{A}_2$ adaptive execution H1-Query. $\mathcal{B}$ saves a list $L_1 \rightarrow \{(ID, h_1, Q)\}$. If ID exist in the list L_1, $\mathcal{B}$ extracts the corresponding list tuple (ID, h_1, Q), and extracts Q to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ randomly selects $h_1 \in Z_p^*$, calculates $Q = g^{h_1}$. $\mathcal{B}$ returns Q to $\mathcal{A}_2$ and inserts (ID, h_1, Q) into L_1.

SecretValue-Query: For identity ID, $\mathcal{A}_2$ adaptive execution Secretvalue-Query. $\mathcal{B}$ saves a list $L_2 \rightarrow \{(ID, pk_{ID}, S_{ID}, \eta)\}$, $\mathcal{B}$ queries the list L_2.

- If ID does not exist in the list L_2, $\mathcal{B}$ randomly selects $x \in Z_p^*$, and throws a coin $\eta \in \{0, 1\}$. Suppose the probability of $\eta = 0$ is ω, then the probability of $\eta = 1$ is $1 - \omega$. When $\eta = 0$, $\mathcal{B}$ calculates $pk_{ID} = g^x$, returns S_{ID} to $\mathcal{A}_2$, and inserts (ID, pk_{ID}, x, η) into the list L_2. When $\eta = 1$, $\mathcal{B}$ calculates $pk_{ID} = (g^a)^x$, inserts into the list, and then $\mathcal{B}$ aborts.
- If ID exists in the list L_2, $\mathcal{A}_2$ queries the corresponding value of η. If $\eta = 1$, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ directly extracts S_{ID} and returns to $\mathcal{A}_2$.

PublicKey-Query: For identity ID, $\mathcal{A}_2$ adaptive execution Publickey-Query.

- If ID does not exist in the list L_2, $\mathcal{B}$ randomly selects $x \in Z_p^*$, and throws a coin $\eta \in \{0, 1\}$. If $\eta = 0$, $\mathcal{B}$ calculates $pk_{ID} = g^x$ and if $\eta = 1$, $\mathcal{B}$ calculates $pk_{ID} = (g^a)^x$. In addition $\mathcal{B}$ returns pk_{ID} to $\mathcal{A}_2$, and inserts (ID, pk_{ID}, x, η) into the list L_2.
- If ID exists in the list L_2, $\mathcal{B}$ directly extracts pk_{ID} and returns it to $\mathcal{A}_2$.

H2-Query: $\mathcal{A}_2$ adaptive execution H2-Query for w. $\mathcal{B}$ saves a list $L_3 \rightarrow \{(w, h_2, y)\}$. If w exists in the list L_3, $\mathcal{B}$ extracts the corresponding y to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ randomly selects $h_2 \in Z_p^*$, evaluats $y = (g^b)^{h_2}$ to $\mathcal{A}_2$ and inserts (w, h_2, y) into the list L_3.

Tag-Query: For identity ID, $\mathcal{A}_2$ adaptive execution Tag-Query. $\mathcal{B}$ queries the list $L_2 \rightarrow \{(ID, pk_{ID}, S_{ID}, \eta)\}$, if $\eta = 1$, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ computes D_{ID} and extracts S_{ID} from the list L_2, y from the list L_3, and calculates the corresponding tag $\sigma_{ij} = (g^{h_1})^{\chi \sum_{k=1}^{s} b_{ijk}} \cdot y^{S_{ID}}$ to $\mathcal{A}_2$.

TABLE 3. Feature comparisons.

Schemes	Certificateless	Multi-cloud	Identity tracking	Data dynamics	Limit TPA
[16]	✗	✓	✗	✗	✗
[17]	✗	✓	✗	✗	✗
[18]	✗	✗	✗	✓	✗
[19]	✗	✓	✗	✗	✗
[20]	✗	✓	✗	✓	✗
[21]	✗	✗	✗	✓	✗
[43]	✗	✓	✗	✗	✗
Ours	✓	✓	✓	✓	✓

Forge: $\mathcal{A}_2$ forges a tag based on identity ID' and its data block b_{ij}. The requested block b_{ij} has not performed a Tag-Query.

Analysis: If $\mathcal{A}_2$ win the game successfully, $\mathcal{B}$ obtain $e(\sigma'_{ij}, g) = e(H_1(ID')^{\sum_{k=1}^{s} b_{ijk}}, mpk) \cdot e(H_2(w'), pk_{ID'})$, further get $e(\sigma'_{ij}, g) = e(g^{h'_1 \sum_{k=1}^{s} b_{ijk}}, g^{\chi}) \cdot e(g^{bh'_2}, g^{ax})$, so we can get $g^{ab} = (\sigma'_{ij})^{1/x'h'_1h'_2\chi \sum_{k=1}^{s} b'_{ijk}}$. Next we analyze the probability that $\mathcal{B}$ not interrupt. From the above analysis, we can get the interrupt only happens in SecretValue-Query and Tag-Query, and the probability of $\mathcal{B}$ not interrupt is $(1-\omega)^{q_s+q_T}$, so the probability of $\mathcal{B}$ win the game in time $t' \leq t + O(q_{h_1} + q_s + q_{pk} + q_{h_2} + q_T)$ is $\varepsilon' \geq \varepsilon \cdot \omega \cdot (1-\omega)^{q_s+q_T} \geq \varepsilon/((q_s+q_T)\cdot 2e)$.

Theorem 2: The cloud server generates the *proof* based on the correct data can pass the verification of TPA.

Proof: If the integrity proof (σ', φ') output by adversary $\mathcal{A}_3$ passes the TPA verification, then there is a challenger $\mathcal{B}$ who can solve the DL problem with a non-negligible probability. Given an example of a DL problem (b_1, b_2), where $b_2 = b_1^x$. $\mathcal{B}$ and $\mathcal{A}_2$ perform the following security game, and calculate x.

Game3: This game is similar to Game 1, with one difference. Challenger $\mathcal{B}$ generates challenge information to $\mathcal{A}_3$. $\mathcal{A}_3$ forges an evidence (σ', φ') to challenger B. The correct proof generated by the cloud server satisfy the following equation: $e(\sigma, g) = e((H_1(ID_u)^{\varphi}, mpk) e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i||t_i||i_\varsigma||j)^{v_{i\varsigma}}, pk_u)$. Assuming that the forged proof of $\mathcal{A}_3$ can be verified by $\mathcal{B}$, we can obtain $e(\sigma', g) = e((H_1(ID_u)^{\varphi'}, mpk) e(\prod_{j=1}^{r}\prod_{\varsigma=1}^{c} H_2(v_i||t_i||i_\varsigma||j)^{v_{i\varsigma}}, pk_u)$.

The above has proved that the tag is not forgeable, therefore $\sigma' = \sigma$. And from the assumption, we know $\varphi' \neq \varphi$. From the above two equations, we can get $H_1(ID_u)^{\varphi'} = H_1(ID_u)^{\varphi}$. Definition $\Delta\varphi = \varphi' - \varphi$, further get $H_1(ID_u)^{\Delta\varphi} = 1$. $\mathcal{B}$ randomly chooses $\alpha, \beta \in Z_q^*$, sets $H_1(ID_u) = \vartheta = b_1^{\alpha} b_2^{\beta}$, gets $\vartheta^{\Delta\varphi} = (b_1^{\alpha} b_2^{\alpha})^{\Delta\varphi} = b_1^{\alpha\Delta\varphi} b_2^{\beta\Delta\varphi} = 1$, $b_2 = h^{\frac{\alpha\Delta\varphi}{\beta\Delta\varphi}}$, further gets $x = \frac{\alpha\Delta\varphi}{\beta\Delta\varphi}$. From the above analysis, we know $\Delta\varphi \neq 0$, $\beta \in Z_q^*$, and the probability of $\beta = 0$ is $1/q$, so the DL problem is solved with a non-negligible probability $1 - 1/q$.

Theorem 3: Our scheme can resist malicious auditors.

Proof: According to the nature of the blockchain, the nonce of the block is unpredictable. We use nonces to generate challenge information, which ensures that the data blocks participating in the challenge cannot be calculated in advance. In addition, transactions recorded on the blockchain are time-sensitive, and it is impossible for TPA to record the audit results on the blockchain at a later time. Therefore, our scheme can resist malicious auditors.

B. PERFORMANCE ANALYSIS

We analyze and compare our scheme with other multi-replica schemes in terms of features, communication and computation overhead in this section. Based on the PBC library version 0.4.7, the simulation experiments of our scheme and schemes [18], [43] are carried out. The experimental environment is Intel Core i5 processor with 8 GB RAM.

To simplify the expression, we utilize $|Z_p|$ to denote the size of an element in $|Z_p|$, $|G_1|$ to denote the size of an element in $|G_1|$, n to denote the number of data blocks, r to denote the number of replicas (the number of CS), and s to denote number of sectors, c to denote the number of data blocks participating in the challenge. *Tmul*, *Texp*, *Tp*, *Thash* and *Tadd* are used to denote the time required for one multiplication operation, one power operation, one bilinear pairing operation, one hash operation and one addition operation respectively.

Table 3 compares of our scheme with schemes [16]–[21] and [43] on the features of Certificateless, Multi-cloud, Identity tracking, Data dynamics and Limit TPA. It can be seen from Table 3 that schemes [16]–[21] and [43] are faced with the problem of certificate management or key escrow, and do not support identity tracking. Furthermore, these schemes all default that TPA is completely trusted. Our scheme adopts certificateless cryptosystem, and realizes identity tracking of malicious users and dynamic updating of cloud data. In addition, we use blockchain technology to limit the behavior of TPA.

TABLE 4. Comparisons of communication cost.

Schemes	Challenge	Proof Generation
[18]	$2c\|Z_p\| + \|G_1\|$	$2rc\|G_1\| + r\|Z_p\|$
[43]	$3\|Z_p\|$	$(s+3)\|G_1\| + s\|Z_p\|$
Ours	$6\|Z_p\| + \|G_1\|$	$\|G_1\| + \|Z_p\|$

Table 4 compares the communication cost of our scheme with scheme [18] and [43] in the challenge-response phase. Our scheme requires $6|Z_p| + |G_1|$ communication overhead

TABLE 5. Comparisons of computation cost.

Scheme	Tag Generation	Proof Generation	Proof Verify
[18]	$nr\,(Thash + 2Texp + Tmul)$	$r(cTexp + 2cTmul + 2cTadd)$	$r(Tp + cThash + Texp + cTmul)$
[43]	$(2nr+s)Texp + nr(s(Tmul+Tadd) + Thash)$	$rc(Texp+2Tmul + Tadd)$	$3Tp + (rc+s)(Texp+Tmul) + cTadd+crThash$
Ours	$nr\,(2Texp + Tmul + sTadd + Thash)$	$r(cTexp + 2cTmul + csTadd)$	$3Tp + rc(Texp + Tmul + Thash)$

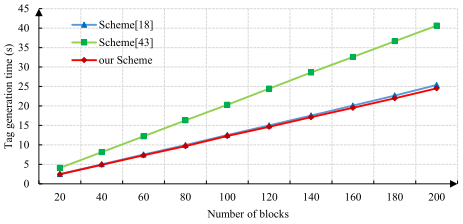

FIGURE 7. Signature efficency.

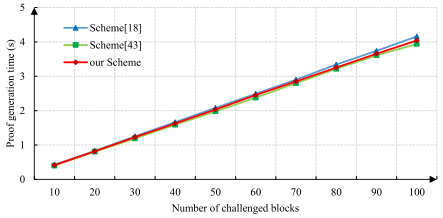

FIGURE 8. Proof generation.

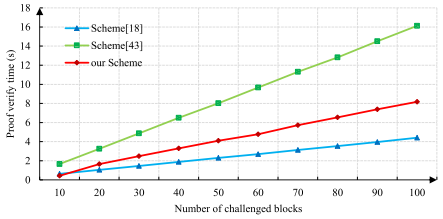

FIGURE 9. Proof verify.

in the challenge phase, and $|G_1| + |Z_p|$ communication overhead in the proof generation phase. Therefore, our scheme has lower communication overhead compared with scheme [18], [43].

Table 5 compares the computation cost of our scheme with scheme [18] and [43] in the stages of tag generation, proof generation and proof verification. In the tag generation phase, the computation cost of scheme [18] is $nr(Thash + 2Texp + Tmul)$ and our scheme is $nr(2Texp + Tmul + sTadd + Thash)$. In the proof verification phase, scheme [43] requires $3Tp + (rc+s)(Texp+Tmul) + cTadd + crThash$computation cost, while our scheme requires $3Tp + rc(Texp + Tmul + Thash)$. Therefore, compared with scheme [18] and [43], our scheme has relatively low computation overhead.

We choose 0-200 data blocks, 3 replicas and 5 sectors to compare the computation cost of our scheme with scheme [18] and [43]. As shown in Fig. 6, our scheme has lower computation cost in the stage of Tag Generation compared with scheme [43]. As shown in Fig. 7, our scheme has similar computation cost with scheme [18] and [43] in the stage of Proof Generation. As shown in Fig. 8, our scheme has lower computation cost with scheme [43]. Compared with scheme [18], when the number of replicas is 3, our efficiency is relatively low. But it can be seen from Table 4 that with the increase of the number of replicas, the cost of scheme [18] in the proof verify stage will increase greatly.

VII. CONCLUSION

We design a multi-replica and multi-cloud data public audit scheme, which not only supports the modification, insertion and deletion of cloud multi-replica data, but also can track the identity of malicious users. In addition, blockchain technology is introduced to restrict the behavior of third-party auditors. The analysis results show that our scheme satisfies the unforgeability of tag and the robustness of audit, and can resist malicious auditors. Compared with the similar scheme, our scheme has higher performance in communication and computation overhead.

REFERENCES

[1] 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 Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[2] 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, Sep. 2017.

[3] J. Li, H. Ye, W. Wang, W. Lou, Y. Thomas, J. Liu, and R. Lu, "Efficient and secure outsourcing of differentially private data publication," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 187–206.

[4] M.-S. Hwang, C.-C. Lee, and T.-H. Sun, "Data error locations reported by public auditing in cloud storage service," *Automated Softw. Eng.*, vol. 21, no. 3, pp. 373–390, Sep. 2014.

[5] C. Cachin, I. Keidar, and A. Shraer, "Trusting the cloud," *ACM SIGACT News*, vol. 40, no. 2, pp. 81–86, Jun. 2009.

[6] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[7] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 415–430, Apr. 2019.

[8] R. Perlman, "An overview of PKI trust models," *IEEE Netw.*, vol. 13, no. 6, pp. 38–43, Nov. 1999.

[9] A. Shamir, "Identity-based cryptosystems and signature schemes," *Adv. Cryptol.*, vol. 196, pp. 47–53, 1984.

[10] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Theory Appl. Cryptol. Inform. Secur.*, Taipei, Taiwan, 2003, pp. 452–473.

[11] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2012, pp. 507–525.

[12] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan. 2014.

[13] Y. Deswarte, J. J. Quisquater, and A. Sadane, "Remote integrity checking," in *Proc. 6th Work. Conf. Integrity Internal Control Inf. Syst.*, 2003, pp. 1–11.

[14] F. Sebe, 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.

[15] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. Int. Workshop Secur.*, 2008, pp. 1–10.

[16] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, Jun. 2008, pp. 411–420.

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

[18] J. W. Li and M. D. Zhu, "MHT-based dynamic data integrity verification and recovery scheme in cloud storage," *J. Comput. Appl.*, vol. 36, no. 7, pp. 2179–2183, 2019.

[19] Z. Hao and N. Yu, "A multiple-replica remote data possession checking protocol with public verifiability," in *Proc. 2nd Int. Symp. Data, Privacy, E-Commerce*, Sep. 2010, pp. 84–89.

[20] Y. Zhang, J. Ni, X. Tao, Y. Wang, and Y. Yu, "Provable multiple replication data possession with full dynamics for secure cloud storage," *Concurrency Comput. Pract. Exper.*, vol. 28, no. 4, pp. 1161–1173, Mar. 2016.

[21] S. Peng, F. Zhou, J. Li, Q. Wang, and Z. Xu, "Efficient, dynamic and identity-based remote data integrity checking for multiple replicas," *J. Netw. Comput. Appl.*, vol. 134, pp. 72–88, May 2019.

[22] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 583–597.

[23] 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.*, 2007, pp. 598–609.

[24] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 676–688, Mar. 2017.

[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, Jul. 2014.

[26] Y. Ren, J. Xu, J. Wang, and J.-U. Kim, "Designated-verifier provable data possession in public cloud storage," *Int. J. Secur. Appl.*, vol. 7, no. 6, pp. 11–20, Nov. 2013.

[27] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1788–1797, Jun. 2020.

[28] C. Erway, A. Kupccu, C. Papamathou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.

[29] Q. Wang, C. Wang, J. Li, K. Ren, and W. J. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2009, pp. 355–370.

[30] 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. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr. 2013.

[31] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 701–714, Sep. 2017.

[32] M. S. Hwang, T. H. Sun, and C. C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *J. Circuits, Syst. Comput.*, vol. 26, no. 5, pp. 175–190, 2017.

[33] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 1, pp. 92–106, Jan. 2015.

[34] Y. Luo, M. Xu, K. Huang, D. Wang, and S. Fu, "Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing," *Comput. Secur.*, vol. 73, pp. 492–506, Mar. 2018.

[35] C.-T. Li, C.-C. Lee, and C.-Y. Weng, "An extended chaotic maps based user authentication and privacy preserving scheme against DoS attacks in pervasive and ubiquitous computing environments," *Nonlinear Dyn.*, vol. 74, no. 4, pp. 1133–1143, Dec. 2013.

[36] C.-T. Li, C.-C. Lee, and C.-Y. Weng, "A secure cloud-assisted wireless body area network in mobile emergency medical care system," *J. Med. Syst.*, vol. 40, no. 5, pp. 117–130, May 2016.

[37] J. Li, H. Yan, and Y. Zhang, "Identity-based privacy preserving remote data integrity checking for cloud storage," *IEEE Syst. J.*, early access, Mar. 18, 2020, doi: 10.1109/JSYST.2020.2978146.

[38] Y, Yu, Y. Mu, J. B. Ni, J. Deng, and K. Huang, "Identity privacy-preserving public auditing with dynamic group for secure mobile cloud storage," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2014, pp. 28–40.

[39] 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.

[40] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Services Comput.*, early access, Jan. 8, 2018, doi: 10.1109/TSC.2018.2789893.

[41] 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.

[42] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.

[43] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, early access, Jul. 16, 2019, doi: 10.1109/TCC.2019.2929045.

[44] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 831–843.

[45] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, p. 32104, Mar. 2019.

[46] Y. Zhang, C. Xu, X. Lin and, and X. S. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, early access, Mar. 29, 2019, doi: 10.1109/TCC.2019.2908400.

[47] H. Tian, F. Nan, H. Jiang, C.-C. Chang, J. Ning, and Y. Huang, "Public auditing for shared cloud data with efficient and secure group management," *Inf. Sci.*, vol. 472, pp. 107–125, Jan. 2019.

[48] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[49] W. G. Ethereum, "A secure decentralised generalised transaction ledger," Ethereum Project, Yellow Paper, 2014, vol. 151, pp. 1–32.

[50] M. Pilkington, "Blockchain technology: Principles and applications," in *Research Handbook on Digital Transformations Cheltenham*. London, U.K.: Edward Elgar Publishing, 2016, pp. 225–253.

[51] V. Buterin, "On public and private blockchains," in *Ethereum Blog*. 2015. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

XIAODONG YANG (Member, IEEE) received the B.S. degree in mathematics from Northwest Normal University, China, in 2002, the M.S. degree in cryptography from Tongji University, China, in 2005, and the Ph.D. degree in cryptography from Northwest Normal University, in 2010. He is currently a Postdoctoral Fellow with the State Key Laboratory of Cryptology of China and a Professor of information and computer science with Northwest Normal University. His research interests include applied cryptography, network security, and cloud computing security. He is also a member of Chinese Cryptology and Information Security Association.

XIZHEN PEI received the B.S. degree from Shanxi Datong University, Datong, China, in 2018. She is currently pursuing the master's degree in computer science with Northwest Normal University. Her current research interest includes cloud computing security.

MEIDING WANG received the B.S. degree from Northwest Normal University, Lanzhou, China, in 2018, where she is currently pursuing the master's degree in computer science. Her current research interest includes cloud computing security.

TING LI received the B.S. degree from Zhengzhou Normal University, Zhengzhou, China, in 2018. She is currently pursuing the master's degree in computer science with Northwest Normal University. Her current research interests include blockchain technology and their applications.

CAIFEN WANG received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2003. She is currently a Professor of computer science with Shenzhen Technology University. Her current research interests include network security, cryptographic protocols, and security engineering. She is also a member of Chinese Cryptology and Information Security Association.