IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Collaborative Auditing Blockchain for Trustworthy Data Integrity in Cloud Storage System

PEI HUANG[1], KAI FAN[1], (Member, IEEE), HANZHE YANG[1], KUAN ZHANG[2], (Member, IEEE), HUI LI[1], (Member, IEEE), AND YINTANG YANG[3], (Member, IEEE)

[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071 China
[2]Department of Electrical and Computer Engineering, University of Nebraska–Lincoln, NE 68588 USA (e-mail: kuan.zhang@unl.edu)
[3]Key Laboratory of Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices, Xidian University, Xi'an, 710071 China (e-mail: ytyang@xidian.edu.cn)

Corresponding author: Kai Fan (e-mail: kfan@mail.xidian.edu.cn).

This work is supported by the National Key RD Program of China (2017YFB0802300), the National Natural Science Foundation of China (No.61772403, U1836203), the Natural Science Foundation of Shaanxi Province (No.2019ZDLGY12-02), the Shaanxi Innovation Team Project (No.2018TD-007), the Xi'an Science and technology innovation plan (No.201809168CX9JC10) and National 111 Program of China B16037.

ABSTRACT Cloud storage system provides data owners with remote storage service, which allows them to outsource data without local storage burden. Nevertheless, the cloud storage service is not fully trustworthy since it may not be honest and remote data would be corrupted. One way to ensure trustworthy preservation of cloud data is the remote data auditing method, through which data owners can check storage reliability of cloud system on demand and avoid potential data corruption in time. However, private auditing methods fail to promise the mutual trust in auditing results. Thus, public auditing methods are introduced, in which traditionally a third party auditor is delegated to interact with cloud service providers for auditing tasks. Although the third party auditor serves as a medium to exchange trust, a centralized third party is hard to stay neutral, which exposes the remote data auditing to some threats such as collusion attacks. To address the trust problem between data owners and cloud service providers, we propose a collaborative auditing blockchain framework for cloud data storage. In this framework, all consensus nodes substitute for the single third party auditor to execute auditing delegations and record them permanently, thereby preventing entities from deceiving each other. Security analysis shows that the proposed framework has advantage of preserving remote data integrity from various attacks. Performance analysis demonstrates that the framework is more functional and resource-friendly than existing schemes.

INDEX TERMS Cloud Storage, Collaborative Blockchain, Public Auditing, Trustworthy Data Integrity

I. INTRODUCTION

WITH the volume of data becoming larger and larger rapidly, cloud storage has made outsourcing data an inevitable trend for resource-constraint data owners including individuals and even organizations. Such cloud service provides not only outsourcing function but also ubiquitous network access and location independence [1], [2], which make data owners not worry about local operating system failures. In the meantime, overheads resulting from data management are greatly reduced, enabling data owners and local devices to focus on data processing.

However, data stored in cloud storage system is out of strong control of its owners, thus suffering from trust problem mainly caused by misbehaviors of the cloud service provider (CSP) [3], [4]. In particular, malicious CSPs may attempt to delete data which is accessed infrequently to save storage space without authorization, or pretend nothing happens when stored data is corrupted. They may even tamper with some data to deceive its owners for other financial profits. The integrity and availability of cloud data are being challenged. Many researchers have developed the remote data auditing (RDA) method for outsourced data to enable data owners to measure the credibility of CSP without further loss. In other words, a data owner who has deleted local copy can still verify the correctness and integrity of remote stored data through the RDA.

To implement RDA, a mechanism of "challenge-proof-verify" is generally adopted between data owner and CSP in private auditing, where the data owner generates data challenges and verifies corresponding proofs from the CSP to track the state of remote data. Nevertheless, the verification process is only executed by the data owner in private auditing. On the one hand, the auditing result may be unfavorable to the CSP intentionally. On the other hand, such private methods bring a lot of burdens on the data owner as data volume and auditing requests increase. To resolve doubt about auditing result, and to make verification process more energy efficient for data owners, a new entity named third party auditor (TPA) is introduced to achieve public auditing, who accepts auditing delegations from data owners and executes them as directed. But such solutions must assume that the TPA behaves in an honest way, which is not a reliable premise in practice. For example, a TPA may collude with either the CSP to hide data corruption or the data owner to deceive for penalty. In addition, an obvious weakness of a single centralized TPA is the single point of failure.

To tackle the mutual trust problem in public auditing, the emerging blockchain enables a decentralized way to track state changes of a system. This technique first proposed by Satoshi Nakamoto [5] is with the features of decentralization, tamper-proof, consistency and traceability. Recent years, a few schemes [6], [7] have combined integrity checking and blockchain-based storage. However, they only regard the blockchain as an immutable ledger, and data owners still verify proofs by itself. In some cases, data owners have to maintain the whole blockchain, which increases much storage burdens.

Hence, inspired by this technique as well, we design a collaborative auditing blockchain (CAB) to enhance mutual trust between data owners and CSPs in the cloud storage system, while reducing as much resource overheads over data owners as possible. In this paper, our main contributions are summarized as follows:

- We design a hierarchical auditing framework to combine RDA and blockchain, which makes all consensus nodes verify data operation records collaboratively and releases data owners from verification cost.
- We propose a credit-based consensus protocol and an incentive mechanism intended to quantify behaviors of entities.
- We extend our work to support some auditing properties such as batch auditing and dynamic auditing.
- We conduct a comprehensive comparison between existing schemes and the proposed scheme. Security analysis and simulation results can meet our design goals.

The remainder of this paper is organized as follows. Section II presents some works related to our work. Section III introduces some preliminaries which serves for auditing process. Section IV is the problem statement including system framework and design goals of the proposed scheme. Section V describes the core of the CAB and auditing protocol. Section VI gives correctness, security, and simulation analysis respectively. In the end, we draw conclusions in Section VII.

II. RELATED WORK

The RDA can be categorized into private auditing and public auditing. The former only contains two types of entities, namely the data owner and the CSP. The auditing process is mainly performed by the data owner. For example, Ateniese et al. [8] first proposed the concept of provable data possession (PDP), which adopted the mechanism of "challenge-proof-verify" to verify the integrity of remote stored data, and has become the basis of many auditing protocols ever since. However, such private protocols increase the burden on the data owner who lacks computing resources. Furthermore, the data owner and the CSP distrust each other in this context. Hence, auditing result may be harmful for the CSP since it can only be obtained from the data owner. To remove the above doubts, Wang et al. [9] first introduced a trusted TPA into PDP to challenge the CSP on behalf of data owners, which implemented public auditing. In this context, delegation of auditing tasks means that the verification process no longer requires the participation of the data owner, thus reducing a lot of computation overheads. Nevertheless, such public auditing schemes did not consider that the single TPA is not always trustworthy and may be bribed by some entity in practice. And it is even more dangerous that the TPA can derive some information about delegators, since all relevant information for verification are transmitted to it.

To address the above problems brought by the TPA in public auditing, several blockchain-based solutions were proposed. Liu et al. [10] replaced TPA with smart contract, where the data owner and the CSP signed an auditing commitment to resist repudiation. Then the data owner could get the hash result of remote data through block identifier, which was compared to the hash previously stored in the blockchain ledger. Obviously, this scheme is not able to resist replay attacks arose from the CSP. Yu et al. [6] presented a fully decentralized data auditing solution without any TPA. They employed homomorphic verifiable tag (HVT) to perform RDA and their solution could effectively resist replay attacks due to random challenge set generated in every auditing request. But the data owner had to traverse the blockchain ledger to find the specific proof for every challenge, which brought extra cost. Yang et al. [11] combined merkle hash tree (MHT) and timestamp server to ensure that all behaviors of data owners and CSPs satisfied accountable traceability. In their design, the data owner should store all proofs and compute the tree root from leaf nodes during every verification process. Qi et al. [12] suggested reputation to quantify the reliability of a CSP, and improved its blockchain with a two-step validation. But more details were not given in this scheme. Li et al. [13] separated operation behaviors and file information within a block. They also introduced a proxy node to efficiently search specific blocks. However, the data owner needed to download the whole file to verify the integrity and could not afford to require auditing from time to time. Xue et al. [7]

utilized the nonce in blockchain to construct unpredictable challenges, thereby preventing malicious TPA from forging auditing results. The TPA would insert all proofs and auditing results into a log file, which is uploaded to the blockchain afterwards. Nevertheless, it still required the data owner to review the log from blockchain, where a subset should be re-verified by the data owner to discover malicious TPA.

In conclusion, existing blockchain-based auditing schemes mostly focus on solving the collusion problem in TPA-based schemes through recording entities' behaviors or removing the TPA. However, they only take the blockchain as an immutable ledger rather than a distributed trust network, and still involve data owners in nearly the whole auditing process, which causes much burden especially on resource-constrained ones. Therefore, to make use of trust that consensus nodes can contribute to the auditing process, we re-describe relationships among entities in the cloud storage system based on the blockchain, and design a CAB for resource-constrained data owners. There is no more need for them to traverse the CAB ledger to obtain auditing results, while various security threats cannot work on the CAB. We also introduce a credit score mechanism to encourage all participants to keep honest and maintain the stability of the CAB. In addition, a new entity called group manager is set up to separate data owners from consensus process, enabling our design to accommodate more data owners.

Additionally, data dynamics support is also a hot spot attracting scholars' attention these years. In 2008, Ateniese et al. [14] improved the PDP and achieved partially dynamics. Later, Wang et al. [15] developed the MHT which has been employed widely to support full data dynamics. And Zhu et al. [16] introduced a structure called as index hash table (IHT), which recorded the changes of data blocks. However, the above schemes required large communication resources during the updating and verification processes. Then in 2016, Tian et al. [17] constructed a single linked sequence table DHT, which reduced the computation cost of the CSP and communication overheads in the updating process. Inspired by the DHT, we also introduce an auxiliary chain table (ACT) to support data dynamics and help data owners and consensus nodes search records in the CAB efficiently.

III. PRELIMINARIES

Let G be an elliptic curve subgroup, and G_T be a multiplicative subgroup. They are of a large prime order p, and g is a generator of G.

A. COMPUTATIONAL DIFFIE-HELLMAN PROBLEM

A computational problem generated with a security parameter λ is hard if, given as input a problem instance, the probability of finding a correct solution to this problem instance in polynomial time is a negligible function of λ. The security of our scheme is based on the hardness of CDH problem, which is at least as hard as discrete logarithm problem (DLP).

Definition 1 (CDH problem) Given g^a and g^b, where a, $b \in_R Z_p^*$, compute g^{ab}.

B. SYMMETRIC BILINEAR PAIRING

Bilinear pairing for scheme construction is built from a pairing-friendly elliptic curve where it should be easy to find an isomorphism from the elliptic curve group to the multiplicative group. It is a relatively mature and efficient method and has been employed in cloud auditing. The definition of symmetric pairing is stated as follows.

Definition 2 (Symmetric Bilinear Pairing) A map function $e : G \times G \rightarrow G_T$ is a symmetric bilinear pairing only when it satisfies three properties below:

- Bilinear: For $\forall u$, $v \in G$, a, $b \in Z_p$, there is $e(u^a, v^b) = e(u,v)^{ab}$.
- Non-Degeneracy: $e(g, g)$ is a generator of G_T.
- Computability: For $\forall u$, $v \in G$, there exists efficient algorithms to compute $e(u, v)$.

C. BLS-BASED HOMOMORPHIC VERIFIABLE TAG

BLS-HVT is based on the BLS signature algorithm, which can be efficiently aggregated and verified without disclosing private key.

Definition 3 (BLS-HVT) Given a data block blk, and a cryptographic hash function $H : \{0,1\}^* \rightarrow G$. Select a random secret key $sk = a \in Z_p^*$, and compute the corresponding public key $pk = g^a$. Then the BLS-HVT for blk is $\sigma_{blk} = H(blk)^a$. For verification, the verifier will simply need to check whether $e(\sigma_{blk}, g) = e(H(blk), pk)$ holds.

With BLS-HVT, blockless verifiability can be realized.

IV. PROBLEM STATEMENT

A. SYSTEM FRAMEWORK

As shown in Fig. 1, the proposed framework contains four entities: private key generator (PKG), data owners (DOs), group managers (GMs) and CSPs. In our framework, the PKG is governed by a fully-trusted authority which is responsible for setting public parameters for the whole system and generating key pairs for the GM. The DO is assumed to have limited communication, computation, and storage resources. It generates and sends auditing challenges to the CSP on demand, meanwhile maintaining ACT to tracking changes of data blocks. As a member of DOs, the GM is collectively designated by a certain group of DOs. However, it is assumed to possess more resources than common DOs. The GM is responsible for maintaining the blockchain ledger for managed DOs, and returning results to them when ledger is updated. The CSP provides DOs with significant storage space and computation capability. It is also responsible for storing the chain of blocks, while responding proofs to auditing challenges. In our assumption, the CSP may not be honest, and there exists business competition among different CSPs.

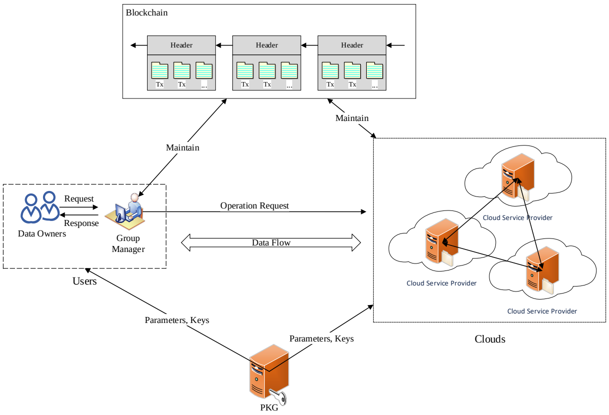

FIGURE 1. The hierarchical CAB framework in cloud storage system.

To support public auditing, the CAB will store auditing requests and proofs, which can be accessed and verified by any entity if need be. In addition, the CAB works more like a permissioned blockchain, since only node who meets some requirements can participate in the consensus process.

As for relationships among the above entities, a brief exposition will be provided here. The DO outsources its data files to a CSP and can retrieve them on demand. When the DO sends an operation request to the CSP through GM, the GM simultaneously broadcasts related auxiliary information to other consensus nodes. Once the CSP confirms request and responds to it, the CAB performs consensus process, and then records this operation and its result in a new block. Eventually the DO is able to obtain final response from the GM after the CAB ledger is updated.

B. THREAT MODEL AND SECURITY ASSUMPTIONS

Generally, an malicious entity in our framework may attack the CAB in three ways:

- by returning positive results to DOs at all times to deceive for rewards, no matter what real responses are.
- by denying requests or operations done to the remote data for further compensation.
- by colluding with other entity to directly interfere consensus judgement.

Since all behaviors are signed and traceable in the CAB, repudiation to what have happened is meaningless. Then we mainly consider four types of attacks in this paper:

- **Replacement attack**. The CSP attempts to pass auditing by replacing the challenged data block and tag with combination of other uncorrupted data blocks and tags.
- **Forgery attack**. The CSP forges proofs to deceive other verifiers.
- **Replay attack**. The CSP replays previous proof which has passed verification to bypass present challenge, or the GM replays an existed and valid auditing result to deceive the DO.
- **Collusion attack**. The CSP colludes with the GM to change challenges and proofs, leading other verifiers to wrong judgement.

We also make some standard cryptographic assumptions. For example, the adversary is not able to forge signatures without owning signer's private key, and the one-way hash function is secure. In addition, we assume that all entities are rational, and no entity can control more than 30% consensus nodes in the CAB.

C. DESIGN GOALS

In this paper, we target public and trustworthy integrity auditing for cloud storage system. The following properties are intended to achieve considering functionality, security and Efficiency respectively.

Functionality:

- **Decentralization**. All operation requests and responses are maintained by all consensus participants so that records can be accessed publicly. Through credit score mechanism, each consensus node has a probability to be in charge of generating new block and writing it into the CAB ledger.
- **Collaboration**. The reliability of auditing result requires all verifiers to devote some resources to execute the verification process.
- **Storage correctness and freshness**. The CSP must pass auditing only if it is storing DO's data files intactly. Furthermore, the CSP keeps the latest version of data blocks and corresponding tags, while the DO and verifiers hold the latest auxiliary information used to verify proofs.
- **Dynamic operations support**. The DO is able to perform remote data update operations, i.e., insertion, deletion, and modification, with necessary cost.

Security:

- **Identity-privacy preservation**. It is impossible for entities except the GM to get the knowledge of DO's identity.
- **Blockless verification**. It enables verifiers to verify proofs without original data, which ensures the security of data to some extent.
- **Unforgeability**. This property indicates that the probability of forging a proof able to pass verification of honest verifier in polynomial time is negligible.
- **Collusion resistance**. It offers the CAB with the ability of resisting against collusion attacks under certain circumstance.

Efficiency:

- **Batch auditing**. Verifiers can check multiple auditing challenges corresponding to various data blocks of different files from different DOs in a management domain simultaneously.
- **Efficient traceability**. All operation histories related to a certain data block can be quickly retrieved by any entity.

TABLE 1. Notations and Descriptions

Notation	Description
λ	a security parameter
G	an elliptic curve group
G_T	a multiplicative group
p	the prime order of G, G_T
g_1, g_2	two random generators of G
$e : G \times G \to G_T$	a bilinear pairing
$H(\cdot) : \{0,1\}^* \to G$	a one-way hash function
$f(\cdot)$	a pseudo-random function
$\pi(\cdot)$	a pseudo-random permutation
Z_p	a field of residue classes modulo p
$\sigma_{k\|\varepsilon}(\cdot)$	a signature signed with key k or by entity ε
ID_ε	the unique identicator of entity ε

- **Stability**. The sharp increase in the number of DOs and GMs will not substantially reduce verification efficiency.

V. THE PROPOSED SCHEME

In this section, we first introduce a dynamic structure to support various operations on data blocks. Then the core of our public auditing scheme, which is based on the hierarchical CAB, is illustrated in detail. Afterwards a description of our auditing protocol is presented in subsection V-C. What follows is the extended property of batch auditing. Finally, the interaction between dynamic data operations and the CAB are described briefly and formally. Table 1 shows some basic notations and their descriptions referred to in our scheme.

A. AUXILIARY CHAIN TABLE

In this part, based on the DHT designed in [17], we develop a new data structure ACT employed by each DO, which provides fast retrieval in the CAB and some metadata that helps verification. The ACT is two-dimensional and its specific structure is shown in Fig. 2. Metadata information in the ACT is categorized into file information and data block information. The left part is the identifier of each file owned by a DO, which forms an array since they are independent in storage order. However, due to the correlation among operations on data blocks within the same file, the right part is designed to be a double linked list, where each item records the current version number, time stamp, last operation height, last operation type, and last operation state of corresponding data block.

In our scheme the version number is denoted as $v_{m,n}$, which is updated when a given data block is inserted or modified. Likewise, the time stamp, last operation height, last operation type, and last operation state are respectively abbreviated to $ts_{m,n}$, $loh_{m,n}$, $lot_{m,n}$ and $los_{m,n}$. We should note that $ts_{m,n}$ and $lot_{m,n}$ are updated when operation request is generated, but $loh_{m,n}$ and $los_{m,n}$ are updated after corresponding responses are obtained from the CAB. Here in the above symbols, m represents the file ID and n represents the index of data block.

With such a double linked array, the insertion and deletion of a data block will cause no change in other items within the same file. Moreover, the ACT provides with local convenience when retrieving state of a certain element or performing forward traceability in the CAB.

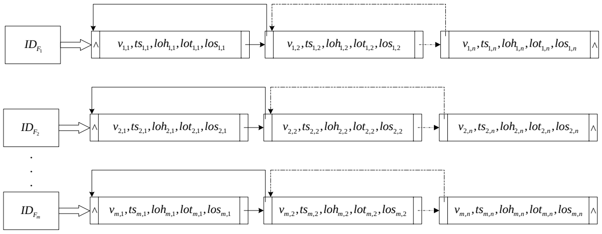
ID_{F_1}
$v_{1,1}, ts_{1,1}, loh_{1,1}, lot_{1,1}, los_{1,1}$
$v_{1,2}, ts_{1,2}, loh_{1,2}, lot_{1,2}, los_{1,2}$
$v_{1,n}, ts_{1,n}, loh_{1,n}, lot_{1,n}, los_{1,n}$
ID_{F_2}
$v_{2,1}, ts_{2,1}, loh_{2,1}, lot_{2,1}, los_{2,1}$
$v_{2,2}, ts_{2,2}, loh_{2,2}, lot_{2,2}, los_{2,2}$
$v_{2,n}, ts_{2,n}, loh_{2,n}, lot_{2,n}, los_{2,n}$
ID_{F_m}
$v_{m,1}, ts_{m,1}, loh_{m,1}, lot_{m,1}, los_{m,1}$
$v_{m,2}, ts_{m,2}, loh_{m,2}, lot_{m,2}, los_{m,2}$
$v_{m,n}, ts_{m,n}, loh_{m,n}, lot_{m,n}, los_{m,n}$

FIGURE 2. The auxiliary chain table for supporting data dynamics.

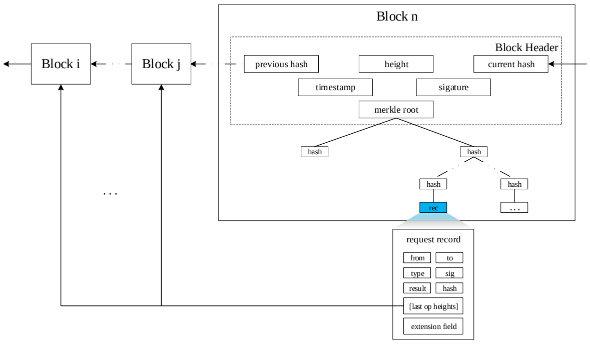

FIGURE 3. The structure of a block in the CAB ledger.

For further operation convenience, we manually categorize the ACT into existing ACT (EACT) and deleted ACT (DACT). When a data block has been deleted, corresponding item would be deleted from the EACT and inserted into the DACT. In this way, the order of data blocks within the same file would not be chaotic when a new data block is to be inserted afterwards.

B. COLLABORATIVE AUDITING BLOCKCHAIN

The main function of the CAB is to encourage participants with credit rewards in packaging operation information and verifying proofs to make auditing results more reliable. In this part, we present the designed CAB from aspects of block structure, consensus process, and incentive mechanism, which are the basis of a blockchain.

1) Block Structure

We should note that every practical blockchain system has a strong requirement for application scenario, and existing security architectures based on the blockchain have their own security goals. Therefore, information stored in a block are different in these systems. In practice, considering that the capacity of a block is limited for the reason of efficiency, we only store the most important metadata which reflects entities' behaviors and corresponding results in the form of hash.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2993606, IEEE Access

As shown in Fig. 3, a block is divided into two parts: block header and block body. Different from cryptocurrency, we adapt the block to remote auditing context by reconstructing transaction structure. Details about important fields are showed as follows:

- **height**: the hight of current block, which is responded to the DO to update the ACT.
- **from, to**: identifiers of delegated GM and requested CSP.
- **type**: an operation type among insertion, modification, deletion and auditing.
- **sig**: the signature of GM who is responsible for transmitting the transaction to the CAB network.
- **result**: the result to operation request signed by the representative.
- **last op heights**: a list of the heights of previous blocks which track last operations done to the same data blocks. It is filled with the field loh obtaining from the ACT.
- **extension field**: if operation type is auditing, this field is filled with challenges and corresponding proofs. If operation type is insertion or modification, this field is written with a list of v and ts obtaining from the ACT. Otherwise it is empty.

2) Consensus Process

Apart from the construction of block structure for specific scenario, another crucial part of a blockchain is the design of its consensus process. Generally, the purpose of a consensus protocol is to solve problems of storage consistency and process reliability. Based on [18]–[20], which are known as PoS and PBFT, we define credit score as measurement of CSPs' and GMs' reliability and a factor affecting selection of consensus nodes and representative. Before describing consensus process, we give definitions about some concepts.

Definition 4 (Credit Score) Credit score cr is employed to express the degree at which a CSP or a GM can be trusted via evaluating their behaviors.

Definition 5 (Credit Block) A special block containing a list of CSPs and GMs identified by their public keys and credit scores $\{(pk_1, cr_1), (pk_2, cr_2), \dots\}$.

On the whole, the credit-based consensus process is run by all CSPs and selected GMs over a sequence of rounds in our framework. Before establishing the CAB, through negotiating, CSPs and GMs get their initial credit scores according to the percentage of DOs that they provides service for or they manages. Then the representative election proceeds as follows:

1) **Initialization**: When system starts, CSPs and GMs broadcast their public keys and credit scores $(pk, cr, \sigma_{PKG}(pk \parallel cr))$. Then all CSPs and GMs get the credit block to initialize the CAB.
2) **Consensus nodes selection**: As the number of DO rises, there would be more GMs for load balancing. Considering efficiency and stability of consensus process, we first sort GMs by the number of request source in current transaction pool in descending order and get $\{GM_1, GM_2, \dots\}$. Then GM_i is selected to be a consensus node in current round with a probability p_i, where $p_i = \frac{cr_i}{\sum_j cr_j}$. Consensus nodes selection process flips a p_1-biased coin to check whether GM_1 is selected; then for all $j \geqslant 2$ if exists, it flips a $(1-p_1)\cdots(1-p_{j-1})p_j$-biased coin to check whether GM_j is selected. Once a GM is picked out, the sorted set of GMs would remove it. The selection process would be executed $2f+1$ times, where $f \in \mathbf{Z}$ is the maximum number of possible malicious GMs in the CAB. Additionally considering that the number of clouds is limited in reality, all CSPs will participate in the consensus process in our scheme.

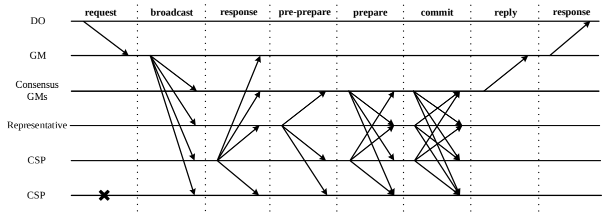

FIGURE 4. The communication flow from DO's request to the end of consensus.

3) **Representative election**: Similar to consensus nodes selection in GMs, except that we first sort CSPs by the number of request destination in current transaction pool in ascending order. After rearranging the set of consensus nodes in the order of $\{\{CSPs\}, \{GMs\}\}$, the p-biased coin selection process will be only executed once to determine current representative.

For every round after the representative has been elected, the abstract flow from sending requests to the end of consensus is as Fig. 4 shows.

As a prerequisite, a DO sends an operation request req on remote stored data to the GM, and GM broadcasts req to the CAB network after re-signing it. Upon receiving req, the designative CSP signs response res and broadcasts to the network as well. When the representative collects enough (req, res) to form a block, it adds its verification result to each (req, res) and launches a three-stage consensus proposal. Firstly, the representative broadcasts a pre-prepare message and a new packaged block. When a consensus node receives the message, it checks correctness of current round information. If passed, this consensus node broadcasts a prepare message to claim its ready state. Once a consensus node obtains more than $2N/3$ prepare messages, where N is the total number of consensus nodes in current round, it begins verification process with relevant information stored in the CAB ledger to check whether the result is the same as what the representative gets or not. If the same, this node will broadcast a commit message to other peers. Finally, if a consensus node gets more than $2N/3$ consensus commits, it will accept the new block and append it to the end of ledger. Then all consensus GMs reply to other unselected GMs in current round, each of which will accept the new block if more than $f + 1$ same blocks are received. Eventually, the GM responds to the DO with operation result after ledger is updated.

3) Incentive Mechanism

Another important part to a blockchain is the incentive mechanism, which can encourage consensus nodes to repeat some tasks honestly and get their rewards, thus enhancing the distributed trust of the whole system and the reliability of results. Based on the assumption that all entities are rational, we roughly present a feasible incentive mechanism that takes economic benefit and misbehaviors into consideration for further stability:

- A DO needs to buy credits to pay GMs and CSPs for operation requests.
- Each communication in consensus process will cost GMs and CSPs a portion of credit.
- A CSP who fails DO's request will decrease exponentially on credit score, whereas the DO gets extra compensation apart from request fee which has been paid.
- A GM can only get credit rewards when managed DO's request passes, otherwise the GM makes neither profit nor loss.
- A representative will get large but linear and other consensus nodes will get small but also linear increase on credit scores if current round successfully reaches final consensus.
- A DO has the freedom to choose any CSP to store data or any GM to transfer messages according to the credit score. If the credit score falls below a threshold, existed consumers will leave for another CSP or GM.

It is obvious that the credit score not only affects the probability of consensus nodes selection and representative election, but also determines practical economical benefit of all entities, since it is like a token to some extent. If a CSP keeps data intact, performs DO's requests honestly, or reports other CSPs' misbehaviors, it will be rewarded with some credit scores. In other words, if a CSP is detected having some misbehaviors, which eventually leads to failing to pass auditing in our scheme, it will be punished hard in credit score. Any DO prefers choosing a CSP which has a relatively high credit score to store data, and such CSPs can also be elected to be the representative with a high probability. Likewise, for a rational GM, though its credit score will not decrease even if DOs' requests fail, a DO tends to send requests to a GM with a relatively high credit score when there are other GMs within a big domain, which further influence the probability of a GM to be selected as a consensus node. Therefore, the proposed incentive mechanism guarantees the honesty of entities to a considerable extent.

C. HIGH DESCRIPTION OF THE AUDITING PROTOCOL

In this part, we focus on our public auditing protocol with the help of the CAB described in subsection V-B. The construction of the proposed protocol is divided into two phase: setup phase and audit phase. The former is responsible for system parameters initialization and auxiliary information generation. The latter is the core process of remote data auditing. Both phases need collaboration among all participants, that is, the interaction with the CAB.

Assuming that a PKG, a DO, a GM which the DO chooses, and all CSPs participate in our auditing protocol. In addition, the specific signature algorithm for message authentication is out of our consideration.

1) Setup Phase

In this phase, the PKG is in charge of system parameters initialization, while the DO and GM need to pre-process files. Firstly, the PKG utilizes security parameter to generate a public-secret key pair for the GM, and the DO also generates its own key pair in **KeyGen**. Then the DO initializes EACT and DACT in **FileToAux** to store metadata in the form of as described in subsection V-A. Finally, the DO pre-processes files and uploads them to the CSP in **FileToCS**.

KeyGen. With a security parameter λ, the PKG first selects an elliptic curve group G and a multiplicative group G_T of the same large prime order p, a field Z_p of residue classes modulo p, and a symmetric bilinear pairing $e : G \times G \rightarrow G_T$. Two random generators $g_1, g_2 \in G$ are also picked. Additionally, the PKG defines a one-way hash function $H : \{0,1\}^* \rightarrow G$, a pseudo-random function (PRF) f, and a pseudo-random permutation (PRP) π. Thus, the set of system public parameters is

$$\mathbb{SP} = \{G, G_T, p, Z_p, g_1, g_2, e, H, f, \pi\}. \tag{1}$$

Then the PKG chooses random element $\alpha \in Z_p^* = Z_p \backslash \{0\}$ as the secret key gsk of GM, and let its public key be $gpk = g_1^{\alpha}$. The GM also chooses a signing key pair $(gssk, gspk)$. As for the DO, it randomly chooses $sk = \beta \in Z_p^*$ as private key, and computes $pk = gpk^{\beta}$ to be public key. Afterwards the DO calculates the inverse inv satisfying

$$inv \cdot \beta \equiv 1 \pmod p,$$

and a parameter $\gamma = gpk^{inv}$. Likewise, the DO chooses a signing key pair (ssk, spk). Furthermore, we assume that the CSP which is designated to offer storage service to the DO has signing keys $(cssk, cspk)$.

FileToAux. Initially the DO creates the EACT for all files to be uploaded. For i-th data block in a file, we set corresponding item as

$$(v_i = 1, ts_i, loh_i = -1, lot_i = insert, los_i = -1)$$

in the EACT.

FileToCS. For simplicity, we suppose that the DO needs to upload a file F. The DO splits F into data blocks such as:

$$F = \{b_1, b_2, \cdots, b_i, \cdots, b_n\} \quad i \in [1, n],$$

where b_i is a general name of data block. Then the DO generates a BLS-HVT for each b_i:

$$\sigma_i = (H(v_i \parallel ts_i) \cdot g_2^{b_i})^{sk}, \tag{2}$$

of which the aggregated set is

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

In (2), v_i and ts_i are obtained from the EACT. Afterwards the DO constructs an insertion instruction

$$\begin{aligned}&\{insert, ts_{ins}, ID_F, F, \sigma, H(F),\\ &ID_{CSP}, \{(v_i, ts_i, loh_i)\}_{i \in [1,n]}, \sigma_{DO}\}\end{aligned}$$

and sends to the GM, where

$$\sigma_{DO} = \sigma_{ssk}(ts_{ins} \parallel \sigma \parallel \{(v_i, ts_i, loh_i)\}_{i \in [1,n]}),$$

and ts_{ins} is the generation time of this instruction. The GM transforms the insertion instruction and delivers

$$\{insert, ts_{ins}, ID_{GM}, ID_F, F, \sigma, H(F), \sigma_{GM}\}$$

to the CSP for the DO, where

$$\sigma_{GM} = \sigma_{gssk}(ts_{ins} \parallel \sigma).$$

In the meantime, the GM creates and broadcasts a insertion request to the CAB network:

$$\begin{aligned}TX_{ins2CAB} = \{&insert, ts_{ins}, ID_{GM}, ID_F,\\ &\{(v_i, ts_i, loh_i)\}_{i \in [1,n]},\\ &ID_{CSP}, \sigma'_{GM}\},\end{aligned} \tag{3}$$

where

$$\sigma'_{GM} = \sigma_{gssk}(ts_{ins} \parallel \{(v_i, ts_i, loh_i)\}_{i \in [1,n]}).$$

Upon receiving insertion instruction, ID_{CSP} verifies σ_{GM} firstly and then checks integrity of file F through $H(F)$. If passed, it stores F and broadcasts a response

$$TX_{res2ins} = \{ID_{CSP}, ts_{ins}, ID_{GM}, ID_F, 1, \sigma_{CSP}\},$$

where

$$\sigma_{CSP} = \sigma_{cssk}(1 \parallel ts_{ins}).$$

When the representative receives both $TX_{ins2CAB}$ and $TX_{res2ins}$, it verifies GM's and CSP's signatures, and writes $(1, \sigma_{rep}(1 \parallel ts_{ins} \parallel ID_{GM} \parallel ID_F)$ into field **result** of a transaction in the new packaged block if passed. Once the representative collects enough transactions, the consensus process will be performed as described in V-B2. In the end, the GM obtains the latest ledger and returns

$$\{ID_F, 1, h, \sigma_{rep}(1 \parallel ts_{ins} \parallel ID_{GM} \parallel ID_F)\}$$

to the DO, where rep is the abbreviation of the representative and h is set as the hight of the latest block in the updated ledger. The DO verifies the representative's signature with local stored ts_{ins}, and overwrites items in the EACT as

$$(v_i = 1, ts_i, loh_i^* = h, lot_i = insert, los_i^* = 1)$$

if passed, which means that the insertion request is performed successfully. At this moment the DO can delete the local copy of file F.

2) Audit Phase

This phase performs remote data integrity auditing. Specifically, challenges are generated from the DO and sent to the CSP in **ChalGen**, after which the CSP computes integrity proofs and broadcasts them to the CAB network in **ProofGen**. Then in **ProofAudit**, consensus nodes check the correctness of proofs with the help of auxiliary metadata previously stored in the CAB ledger. For simplicity, supposing that the DO wants to audit aforementioned file F which has been uploaded.

ChalGen. The DO randomly chooses two generation keys $k_1, k_2 \in Z_p^*$, and picks $z \in \mathbf{Z}^+$ random items from the EACT, eventually obtaining a challenge set

$$chal = \{i, r_i\}_{i \in [1,z]} \tag{4}$$

by calculating

$$i = \pi_{k_1}(l), \; r_i = f_{k_2}(l) \quad l \in [1, z].$$

Meanwhile, the DO updates these items as

$$(v_i = 1, ts_i^* = ts_{aud}, loh_i = h_i, lot_i = audit, los_i = -1),$$

where ts_{aud} is current time. Then the DO signs $chal$ and other auxiliary information, after which it constructs an auditing instruction and sends it to the GM:

$$\begin{aligned}&\{audit, ts_{aud}, \gamma, ID_F, chal,\\ &\{loh_i\}_{i \in [1,z]}, ID_{CSP}, \sigma_{DO}\},\end{aligned}$$

where

$$\sigma_{DO} = \sigma_{ssk}(ts_{aud} \parallel \gamma \parallel chal \parallel \{loh_i\}_{i \in [1,z]}).$$

The GM constructs and re-signs

$$\begin{aligned}TX_{aud2CAB} = \{&audit, ts_{aud}, ID_{GM}, \gamma, ID_F, chal,\\ &\{loh_i\}_{i \in [1,z]}, ID_{CSP}, \sigma_{GM}\},\end{aligned} \tag{5}$$

where

$$\sigma_{GM} = \sigma_{gssk}(ts_{aud} \parallel \gamma \parallel chal \parallel \{loh_i\}_{i \in [1,z]}).$$

Finally it will be broadcast to all participants for future verification.

ProofGen. Once ID_{CSP} gets the $TX_{aud2CAB}$, it verifies GM's signature and checks whether $chal$ are out of range or not. If valid, ID_{CSP} generates a tag proof

$$TP = \prod_{i \in [1,z]} \sigma_i^{r_i} \tag{6}$$

and a data block proof

$$DP = \sum_{i \in [1,z]} b_i \cdot r_i \tag{7}$$

according to $chal$. Upon completion, the above proofs

$$\begin{aligned}TX_{res2aud} = \{&ID_{CSP}, ts_{aud}, ID_{GM},\\ &ID_F, TP, DP, \sigma_{CSP}\}\end{aligned} \tag{8}$$

will be broadcast, where

$$\sigma_{CSP} = \sigma_{cssk}(ts_{aud} \parallel TP \parallel DP).$$

ProofAudit. The representative collects $TX_{aud2CAB}$ and $TX_{res2aud}$. Based on loh_i, the representative retrieves field **last op heights** of a transaction in corresponding blocks recursively till the type of transaction is "insert" or "modify", and gets all v and ts of challenged data blocks from the CAB ledger. Then the representative computes aggregated data block information

$$DBI = e(\prod_{i \in [1,z]} H(v_i \parallel ts_i)^{r_i}, gpk) \tag{9}$$

for the whole $chal$. And it continues to check whether

$$e(TP, \gamma) = DBI \cdot e(g_2^{DP}, gpk) \tag{10}$$

holds. If holding, the representative writes $(1, \sigma_{rep}(1 \parallel ts_{aud} \parallel chal \parallel ID_{GM} \parallel ID_F)$ into field **result** of corresponding transaction, otherwise it writes $(0, \sigma_{rep}(0 \parallel ts_{aud} \parallel chal \parallel ID_{GM} \parallel ID_F)$. When enough transactions are collected and processed in such way, the representative performs the consensus process. Finally, the GM receives new block and updates local ledger, after which it returns result

$$\begin{aligned}&\{ID_F, state, h,\\ &\quad \sigma_{rep}(state \parallel ts_{aud} \parallel chal \parallel ID_{GM} \parallel ID_F)\}\end{aligned}$$

to the DO. The DO verifies it with stored ts_{aud}, k_1 and k_2, and updates corresponding items in the EACT as:

$$(v_i, ts_i^*, loh_i^* = h, lot_i = audit, los_i = state).$$

If $state = 1$, the challenged data blocks are complete; otherwise if $state = 0$, there is at least one of these outsourced data blocks suffering corruption.

D. BATCH AUDITING

There are many different DOs to be served and multiple auditing tasks may be waiting for processing simultaneously. The batch auditing in our work supports aggregating challenges of multiple files from various DOs managed by the same GM, which can reduce computation overheads of verifiers in **ProofAudit** compared with individual auditing.

Supposing that the $u_i \in [1, u_z]$ DOs delegate auditing tasks of $i \in [1, z]$ data blocks in $f_i \in [1, f_z]$ files to the GM at the same time. Then in **ChalGen**, the DO needs to extends fields $chal$ and $\{loh\}$ in auditing instruction. The batch verification equation is

$$\prod_{u_i=1}^{u_z} e(\prod_{f_i=1}^{f_z} TP, \gamma) \overset{?}{=} DBI' \cdot e(g_2^{\sum_1^{u_z} \sum_1^{f_z} DP}, gpk), \tag{11}$$

where

$$\begin{aligned} DBI' &= \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} DBI \\ &= \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} e(\prod_{i=1}^{z} H(v_i \parallel ts_i)^{r_i}, gpk) \\ &= e(\prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} \prod_{i=1}^{z} H(v_{u_i f_i i} \parallel ts_{u_i f_i i})^{r_{u_i f_i i}}, gpk) \end{aligned}$$

If it holds, the completeness and correctness of all verified files can be ensured, otherwise at least one file of a certain DO is corrupted.

E. DYNAMIC AUDITING

Data flow during insertion and auditing processes has been respectively described in detail. Hence, we proceed to introduce how the other two operations generate trails in the CAB, which are modification ($TX_{mod2CAB}$) and deletion ($TX_{del2CAB}$). In the following, we suppose that a remote data block b_i in file ID_F of a DO will be manipulated. In addition, corresponding item in current EACT is assumed to be:

$$(v_i, ts_i, loh_i, lot_i, los_i)$$

1) Modification. Assuming that b_i is modified to b_i^*. Firstly, the DO updates corresponding item in the EACT to:

$$(v_i^* = v_i + 1, ts_i^*, loh_i, lot_i^* = modify, los_i^* = -1),$$

and generates a new BLS-HVT

$$\sigma_i^* = (H(v_i^* \parallel ts_i^*) \cdot g_2^{b_i^*})^{sk}$$

for the modified data block. Then a modification instruction

$$\{modify, ts_{mod}, ID_{GM}, ID_F, i, b_i^*, \sigma_i^*, H(b_i^*), \sigma_{GM}\}$$

is sent to the CSP with the help of the GM, where ts_{mod} is the generation time of this instruction. After the GM broadcasts

$$\begin{aligned} TX_{mod2CAB} = \{&modify, ts_{mod}, ID_{GM}, ID_F, i,\\ &ID_{CSP}, (v_i^*, ts_i^*, loh_i), \sigma'_{GM}\}, \end{aligned} \tag{12}$$

and the CSP modifies b_i as indicated and broadcasts response

$$TX_{res2mod} = \{ID_{CSP}, ts_{mod}, ID_{GM}, ID_F, 1, \sigma_{CSP}\},$$

the CAB collects them to verify. If their information matches and the CAB ledger is updated after all processes,

$$\{ID_F, i, 1, h_i, \sigma_{rep}(1 \parallel ts_{mod} \parallel ID_{GM} \parallel ID_F)\}$$

will be returned from the GM. The last step for the DO is to validate the result, and overwrite corresponding item to $loh_i^* = h_i$ and $los_i^* = 1$.

2) Deletion. Similarly, after setting $ts_i^* = ts_{del}$, $lot_i^* = delete$ and $los_i^* = -1$, where ts_{del} is the deletion time, a deletion request

$$\begin{aligned} TX_{del2CAB} = \{&delete, ts_{del}, ID_{GM}, ID_F,\\ &i, \sigma_{GM}\} \end{aligned} \tag{13}$$

is broadcast. Once the CSP accepts the request, it makes a response

$$TX_{res2del} = \{ID_{CSP}, ts_{del}, ID_{GM}, ID_F, 1, \sigma_{CSP}\}.$$

Then after $TX_{del2CAB}$ and $TX_{res2del}$ have been written into the CAB, the DO finally gets

$$\{ID_F, i, 1, h_i, \sigma_{rep}(1 \parallel ts_{del} \parallel ID_{GM} \parallel ID_F)\}$$

and updates the EACT as $loh_i^* = h_i$ and $los_i^* = 1$, after which corresponding item will be moved to the DACT.

We can note that los in relevant items would be reset and overwritten at the beginning and the end of each request. This field keeps the state of lot and it has three different values. When los equals to -1, it means that request lot has not gained a response yet. If los is set to 0, which indicates request lot fails, there exists two possible cases: 1) when $lot = audit$, then the outsourced data is not complete or tampered with; 2) otherwise the asked CSP refused to perform request lot for some reasons such as illegal remote data operation. In this case, the CSP should present relevant evidence to other consensus nodes to validate. If $los = 1$, it shows that request lot has been successfully admitted by the CSP or the auditing result is positive.

VI. EVAULATION

In this part, analysis of correctness, security and performance are showed.

A. CORRECTNESS ANALYSIS

There are two primary equations employed in our verification process: (10) for a single auditing task from a single DO and (11) for several auditing tasks from multiple DOs. Supposing that each consensus node in our protocol has received challenges and valid proofs, we will demonstrate the equality of these two equations to prove the correctness of audit phase.

(10) can be derived as follows:

$$\begin{aligned} e(TP, \gamma) &= e(\prod_{i\in[1,z]} \sigma_i^{r_i}, gpk^{inv}) \\ &= e(\prod_{i\in[1,z]} (H(v_i \parallel ts_i) \cdot g_2^{b_i})^{sk\cdot r_i}, gpk^{inv}) \\ &= e((\prod_{i\in[1,z]} H(v_i \parallel ts_i)^{r_i}) \cdot g_2^{\sum_{i\in[1,z]} b_i \cdot r_i}, gpk) \\ &= e(\prod_{i=1}^{z} H(v_i \parallel ts_i)^{r_i}, gpk) \cdot e(g_2^{\sum_1^z b_i \cdot r_i}, gpk) \\ &= DBI \cdot e(g_2^{DP}, gpk). \end{aligned}$$

Based on (10), (11) explains how a verifier aggregates different challenges and proofs while executing less pairings:

$$\begin{aligned} \prod_{u_i=1}^{u_z} e(\prod_{f_i=1}^{f_z} TP, \gamma) &= \prod_{u_i=1}^{u_z} e(\prod_{f_i=1}^{f_z} \prod_{i=1}^{z} \sigma_{u_i f_i i}^{r_{u_i f_i i}}, gpk^{inv_{u_i}}) \\ &= \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} e(\prod_{i=1}^{z} \sigma_{u_i f_i i}^{r_{u_i f_i i}}, gpk^{inv_{u_i}}) \\ &= \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} DBI \cdot e(g_2^{DP}, gpk) \\ &= \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} DBI \\ &\quad \cdot \prod_{u_i=1}^{u_z} \prod_{f_i=1}^{f_z} e(g_2^{\sum_1^z b_{u_i f_i i} \cdot r_{u_i f_i i}}, gpk) \\ &= DBI' \cdot e(g_2^{\sum_1^{u_z} \sum_1^{f_z} DP}, gpk). \end{aligned}$$

We should note that only DOs who are managed by the same GM are able to reduce pairing operations through information aggregation, compared to individual auditing.

B. SECURITY ANALYSIS

In this subsection, we analyze how the CAB resist four types of attacks which could happen during the whole audit phase, including replacing attacks, forgery attacks, replay attacks and collusion attacks. It is noteworthy that in our assumption all other entities apart from the mentioned in each kind of security threat will comply with auditing regulations.

Theorem 1 The CAB can resist replacing attacks generated by the CSP. A malicious CSP is not able to get the correct combination of intact data block information to replace corrupted data block when generating proofs and deceive verifiers.

Proof: Supposing that a data block b_j to be checked has been corrupted but that two data blocks b_{j_1} and b_{j_2}, whose HVT are σ_{j_1} and σ_{j_2} respectively, are well maintained in the CSP. To retrieve the HVT of b_j, the CSP should find out the correct combination of σ_{j_1} and σ_{j_2}. Since

$$\begin{aligned} \sigma_{j_1} &= (H(v_{j_1} \parallel ts_{j_1}) \cdot g_2^{b_{j_1}})^{sk}, \\ \sigma_{j_2} &= (H(v_{j_2} \parallel ts_{j_2}) \cdot g_2^{b_{j_2}})^{sk}, \end{aligned}$$

then the CSP sets that

$$\begin{aligned} \sigma_j^* &= \sigma_{j_1}^{\alpha_{j_1}} \cdot \sigma_{j_2}^{\alpha_{j_2}} \\ &= ((H(v_{j_1} \parallel ts_{j_1}) \cdot g_2^{b_{j_1}})^{sk})^{\alpha_{j_1}} \\ &\quad \cdot ((H(v_{j_2} \parallel ts_{j_2}) \cdot g_2^{b_{j_2}})^{sk})^{\alpha_{j_2}} \\ &= ((H(v_{j_1} \parallel ts_{j_1})^{\alpha_{j_1}} \cdot H(v_{j_2} \parallel ts_{j_2})^{\alpha_{j_2}}) \\ &\quad \cdot g_2^{\alpha_{j_1} \cdot b_{j_1} + \alpha_{j_2} \cdot b_{j_2}})^{sk} \end{aligned}$$

where $\alpha_{j_1}, \alpha_{j_2} \in Z_p$. Comparing σ_j^* and σ_j, it follows that

$$\begin{aligned} &\alpha_{j_1} \cdot b_{j_1} + \alpha_{j_2} \cdot b_{j_2} = b_j, \\ &H(v_{j_1} \parallel ts_{j_1})^{\alpha_{j_1}} \cdot H(v_{j_2} \parallel ts_{j_2})^{\alpha_{j_2}} = H(v_j \parallel ts_j) \end{aligned}$$

must be satisfied simultaneously. Solution to the second equation refers to the DLP on elliptic curve, which means it is hard to work out the desired α_{j_1} and α_{j_2}. Thus the probability that the CSP performs such replacing attacks successfully is negligible. ■

Theorem 2 The CAB can resist forgery attacks from the CSP. A malicious CSP cannot directly forge a tag proof to make (10) hold.

Proof: If the CSP has modified data block b_i to $b_i + off_i$ for $i \in [1, z]$, where off_i represents the modification part.

Then in **ProofGen**, the CSP should have computed the new TP^* which adapts to the new DP^* as follows:

$$\begin{aligned} TP^* &= \prod_{i\in[1,z]} ((H(v_i \parallel ts_i)\cdot g_2^{b_i+off_i})^{sk})^{r_i} \\ &= \prod_{i\in[1,z]} ((H(v_i \parallel ts_i)\cdot g_2^{b_i})^{sk}\cdot g_2^{off_i\cdot sk})^{r_i} \\ &= \prod_{i\in[1,z]} \sigma_i^{r_i}\cdot g_2^{sk\cdot\sum_{i\in[1,z]} off_i\cdot r_i} \\ &= TP\cdot g_2^{sk\cdot\sum_{i\in[1,z]} off_i\cdot r_i} \end{aligned}$$

Since the CSP only owns TP, it must continue to multiply the rest to get TP^*. Note that sk is just the private key of the DO, which cannot be known by any other entities in our assumption. Hence, the CSP could not forge proofs to deceive other verifiers. ■

Theorem 3 During the audit phase, the CAB can resist replay attacks from the CSP or the GM. It can be categorized into two cases: a malicious CSP could not pass verification if it responds to verifiers with previous valid proofs; a malicious GM is not able to deceive the DO successfully even if it returns previous valid auditing result.

Proof: For the former case, the replay attack may be executed by a malicious CSP when it has illegally changed the state of a data block b_j. We assume that b_j's proof information is replaced by its former data block b_j^-. Then the CSP calculates the new proof (TP^*, DP^*) as follows:

$$\begin{aligned} TP^* &= \prod_{i\in[1,j-1]\bigcup[j+1,z]} \sigma_i^{r_i}\cdot\sigma_{j^-}^{r_j}, \\ DP^* &= \sum_{i\in[1,j-1]\bigcup[j+1,z]} b_i\cdot r_i + b_{j^-}\cdot r_j. \end{aligned}$$

Upon receipt, verifiers will analyze proofs according to (10). The left part concerning the tag proof can be written as:

$$\begin{aligned} e(TP^*,\gamma) &= e(\prod_{i\in[1,z]\setminus\{j\}} \sigma_i^{r_i}\cdot\sigma_{j^-}^{r_j}, gpk^{inv}) \\ &= e(\prod_{i\in[1,z]\setminus\{j\}} (H(v_i \parallel ts_i)\cdot g_2^{b_i})^{sk\cdot r_i} \\ &\quad\cdot (H(v_{j^-} \parallel ts_{j^-})\cdot g_2^{b_{j^-}})^{sk\cdot r_j}, gpk^{inv}) \\ &= e(\prod_{i\in[1,z]\setminus\{j\}} (H(v_i \parallel ts_i)\cdot g_2^{b_i})^{r_i} \\ &\quad\cdot (H(v_{j^-} \parallel ts_{j^-})\cdot g_2^{b_{j^-}})^{r_j}, gpk) \\ &= e(\prod_{i\in[1,z]\setminus\{j\}} H(v_i \parallel ts_i)^{r_i} \\ &\quad\cdot H(v_{j^-} \parallel ts_{j^-})^{r_j}, gpk) \\ &\quad\cdot e(g_2^{\sum_{i\in[1,z]\setminus\{j\}} b_i\cdot r_i + b_{j^-}\cdot r_j}, gpk). \end{aligned}$$

Since verifiers would compute DBI by collecting necessary information from the CAB, therefore the right part concerning the data block proof can be expanded as:

$$\begin{aligned} DBI\cdot e(g_2^{DP^*}, gpk) &= e(\prod_{i\in[1,z]} H(v_i \parallel ts_i)^{r_i}, gpk) \\ &\quad\cdot e(g_2^{\sum_{i\in[1,z]\setminus\{j\}} b_i\cdot r_i + b_{j^-}\cdot r_j}, gpk) \\ &= e(\prod_{i\in[1,z]\setminus\{j\}} H(v_i \parallel ts_i)^{r_i} \\ &\quad\cdot H(v_j \parallel ts_j)^{r_j}, gpk) \\ &\quad\cdot e(g_2^{\sum_{i\in[1,z]\setminus\{j\}} b_i\cdot r_i + b_{j^-}\cdot r_j}, gpk). \end{aligned}$$

Comparing the above two expanded equations, we conclude that only when $H(v_j \parallel ts_j) = H(v_{j^-} \parallel ts_{j^-})$, i.e., $v_j = v_{j^-}$ and $ts_j = ts_{j^-}$, can (10) hold. To our knowledge, version number and timestamp of two different data blocks can never be the same. Moreover, it is impossible for the CSP to tamper with DBI, since this variable is calculated by all verifiers with version number and timestamp recorded in the CAB. Hence, replay attacks cannot work in this case.

For the latter case, the replay attack may happen when a malicious GM is greedy for rewards and attempts to deceive the DO with previous result even if the CSP has failed auditing request. However, this evil intention can be easily prevented owing to the representative's signature of ts_{aud}, ID_{GM}, and ID_F contained in the result. These three variables act as a random and unique identifier set by the DO and thus the signature of representative changes every request. Therefore, previous auditing result cannot work any more in this case.

In conclusion, our scheme can resist replay attacks. ■

Theorem 4 The CAB can resist collusion attacks from CSPs and GMs. Colluding CSP and GM cannot deceive the DO by tampering with challenges and adapting proofs to bypass corrupted data blocks.

Proof: In this situation we consider that the CSP and GM have more power to control a part of auditing process. Since the GM is responsible for transferring challenges to the CAB network for DOs, there is a possibility that the GM negotiates with the challenged CSP in advance and then the GM tampers with the range of a challenge set, i.e., $chal \to chal^* = \{(i^*, r_i)\}$ and $\{loh_i \to loh_i^*\}$, to pick out those still intact data blocks so as to avoid auditing failure. Obviously, altered challenge and proof can easily pass other verifiers' verification afterwards. Nevertheless, similar to the method of preventing replay attacks, in **ProofAudit**, our design requires a signed auditing result from the GM, which contains the CAB's acknowledgement of current challenge set. Hence, the DO can judge whether current auditing request has been suffered from collusion attacks based on k_1 and k_2 stored temporarily, which can be used to re-generate $chal$. So our scheme can resist collusion attacks. ■

Theorem 5 Apart from the GM managing a DO, all other verifiers cannot get information about the relation between the DO and challenged data blocks.

Proof: Before broadcast to the CAB network, each request from the DO will be re-constructed by the GM, i.e., removing signature of the DO. What other entities can get is some auxiliary information about a data block. During the audit phase, an extra variable other verifiers will get is $\gamma = gpk^{inv}$ sent along with $chal$. Although there holds an equation $gpk^{inv \cdot sk} = gpk$, verifiers still cannot solve out $gpk^{sk} = pk$ with the help of γ and gpk due to the CDH problem. That is, verifiers are not able to determine which DO in the domain sends the request. Hence, the CAB can protect DO's identity. ■

With respect to common data operations, our CAB satisfies the properties of non-repudiation and accountable traceability. On the one hand, each behavior is signed with unique file information and timestamp, which means that no entity can deny what has happened or reuse previous information. On the other hand, all related records in the CAB ledger can be found rapidly and can directly reflect modification history of a data block.

Lastly, Table 2 summarizes security comparison results of our scheme with several blockchain-based auditing schemes.

C. PERFORMANCE ANALYSIS

In this subsection, we evaluate the performance of the CAB from two aspects: property comparison and computation cost comparison.

Considering functionality and efficiency, Table 3 lists out comparison result in several properties mentioned in subsection IV-C. Our scheme provides all verifiers with common knowledge, where no sensitive information is involved, and they all have to cost some resources to check auditing proofs, which contributes to the reliability of results. As for compared schemes, their consensus members own no prior knowledge and are only responsible for the verification of signatures, which means that the DO still has to execute auditing process by itself. The CAB supports dynamic data due to operations on data blocks instead of on the whole file. In addition, the proposed ACT helps the DO and consensus paticipants efficiently locate related blocks. With respect to storage freshness, [6], [10], [11] cannot prevent all types of mentioned attacks to ensure the integrity of remote data, thus failing this property's requirement. Apart from [6] and ours, other schemes can only process auditing requests and responses in serial order, which means that they do not support batch auditing to reduce the number of calculation. Moreover, blockchain-based designs generally allow permissible entities to search records in the ledger. We add field **last op heights** bound to operation history of a data block to accelerate retrieving, whereas other schemes must iterate through the whole ledger to find out the correct transaction in the worst case. The last property that proposed scheme provides is described as stability, which is influenced by the number of DOs in our consideration. Our CAB meets this target due to the hierarchical structure which is introduced into the consensus process, where the GM is delegated to participate in verification process. Therefore, a sharp increase of DOs will not substantially reduce the efficiency.

The comparison of entities' computation cost with [6] and [7], which also employ HVT for auditing, is shown in Table 4. Computation overheads are mainly distributed in **FileToCS**, **ProofGen** and **ProofAudit** in the comparison. Firstly in **FileToCS**, the DO generates tags for all data blocks to be uploaded. Then in **ProofGen**, the CSP computes proofs according to challenges. Finally in **ProofAudit**, the DO in [6], the TPA and DO in [7], or each consensus node in the proposed scheme verifies the correctness of proofs. From Table 4, we can see that the DO in the CAB costs much less, since the burdens caused by local trust in auditing result is spread across the CAB network, which is adapted for resource-constrained DOs.

Additionally, we evaluate performance of the CAB by conducting several experiments using JAVA SE 8.0 on Ubuntu 16.04 Virtual Machine equipped with Intel Core i5 CPU at 2.3GHz and 4GB RAM. All pairing related calculations are implemented with JPBC library v2.0.0 and type A pairing parameters, in which the group order is set to 160 bits and the base field order is 512 bits. We divided a file into 10,000 shards and the size of each data block is set to 4KB, while the proportion of corrupted data blocks is set to 0.001. According to the loss function theory described in [21], overall considering the optimal balance of a high detection probability and low validation cost, only a limited number of data blocks need to be challenged. Hence, the sample size is changed from 50 to 500 data blocks in our experiments.

Fig. 5 shows the computation cost on the DO side during the whole auditing process. It is obvious that as the sample size increases, the growth rate of DO's computation time in our scheme is almost half that of other two schemes. In fact apart from tag generation, DOs in comparative schemes have to search blockchain ledger to get proofs and execute verification, whereas the DO in proposed scheme does not concern about these processes and can obtain trustworthy results at last.

Fig. 6 shows the average verification time which the DO in [6], the DO and the TPA in [7] and the consensus node in our scheme spend respectively. Thanks to less expensive operations such as map-to-point hash and pairing, and quick search for specific authentication information stored in the CAB with the help of field **last op height**, our verification cost is more acceptable to each involved verifier compared with other schemes.

In Fig. 7, on the one hand when the size of a challenge set is fixed, the total consensus time in a round, which includes **ProofGen** and **ProofAudit**, does not vary a lot as the number of GMs increases. This is because that the actual number of GMs which are to participate in consensus process depends on the stability of current framework, i.e., only a few when most of GMs are honest, which means the total number of consensus nodes is limited and thus the CAB can support more domains without affecting efficiency greatly. On the other hand, when the size of a challenge set becomes bigger,

TABLE 2. Comparison of security properties

Schemes	replacing attack resistance	forgery attack resistance	replay attack resistance	collusion attack resistance	identity privacy protection
[10]	N	Y	N	Y	N
[6]	N	Y	Y	Y	N
[11]	Y	Y	N	Y	N
[7]	Y	Y	Y	Y	N
CAB	Y	Y	Y	Y	Y

TABLE 3. Property comparison in functionality

Schemes	Collaboration	Dynamic support	Storage freshness	Batch auditing	Efficient traceability	Stability
[10]	N	N	N	N	N	Y
[6]	N	Y	N	Y	N	N
[11]	N	N	N	N	N	N
[7]	N	N	Y	N	N	Y
CAB	Y	Y	Y	Y	Y	Y

TABLE 4. Comparison of computation cost

Schemes	DO	CSP	TPA	Consensus Node
[6]	$(n+z)H+(2n+z+3)E+3P+(n+z+1)M$	$H+(z+1)E+2zM$	-	-
[7]	$(2n+2z'+1)H+3P+(6n+3z'+4)M$	$(3z+1)M$	$(2z+1)H+3P+(3z+4)M$	-
CAB	$nH+2nE+nM$	$zE+(2z-1)M$	-	$zH+(z+1)E+3P+zM$

† H: Hash function mapping a string to a point on G.
† E: Modular exponentiation on G.
† P: Bilinear pairing.
† M: Point multiplication on the group.
† n: The total number of data blocks outsourced.
† z: the number of challenged data blocks.
† z': the size of a subset of challenged data blocks.
† -: no such entity in the scheme.

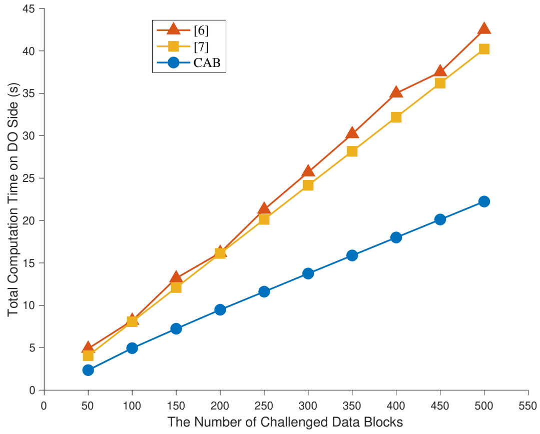

FIGURE 5. The computation cost comparison on the DO during the whole auditing process.

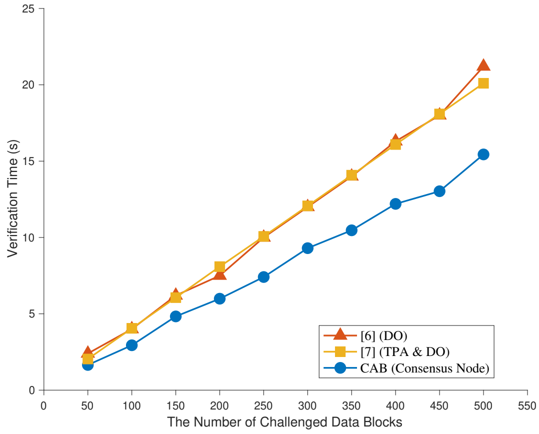

FIGURE 6. The verification cost comparison for each verifier.

it takes more time to get a final auditing result due to the growing waiting latency caused by more proof generation and verification cost. However, it also means that when the amount of data grows, more resources are needed to exchange for more reliable results.

In our scheme, challenges and proofs are only relevant to the GM, and all DOs are separated from consensus process. Therefore, the burdens of consensus nodes barely change when the number of DOs in a group increases from 50 to 500 in Fig. 8, where a challenge is set to the size of 250 data blocks.

Lastly in Fig. 9, we present the comparison of batch

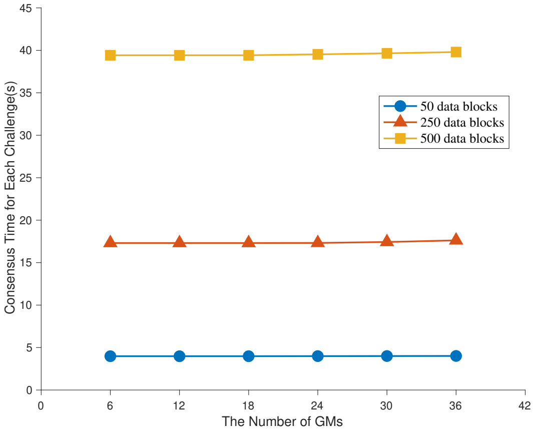

FIGURE 7. The consensus time variation with the number of GMs.

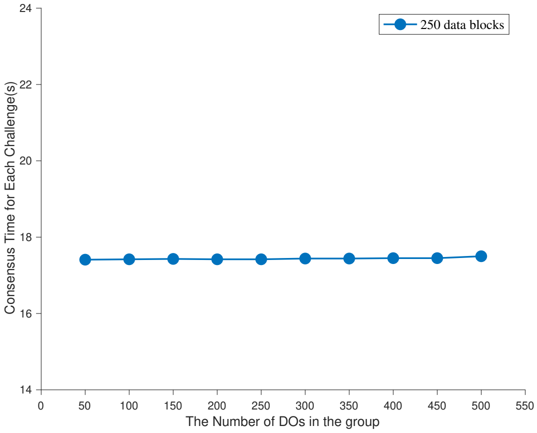

FIGURE 8. The consensus time variation with the number of DOs in a group.

auditing with [6] under the condition that each DO executes auditing tasks on 10 different files, of which each challenge set contains 250 data blocks, and these DOs are from the same group. We can see that as auditing tasks come from more DOs, our average auditing efficiency improves whereas [6] stays nearly unchanged. This is because that the total time cost does not increase linearly when different proofs from multiple DOs are aggregated into (9). However, in the compared work, each DO verifies proofs with local information itself and there is no way to aggregate proofs from various DOs.

VII. CONCLUSION

In this paper, we propose a collaborative auditing scheme based on blockchain to mainly achieve trustworthy data integrity in cloud storage system. By introducing the CAB, the RDA process and its results can be more reliable with the help of interested entities, instead of relying on a single third party. Meanwhile, the proxy role played by the GM separates the DO from consensus process, which means that the DO is free from verification burdens and its identity can be hidden. Moreover, our distributed consensus protocol avoids centralization, and incentive mechanism enhances security and stability of the CAB with the credit score. Security analysis demonstrates that our scheme can handle with multiple security threats. Performance evaluation indicates that the CAB is more resource friendly towards DOs and more functional compared with other blockchain-based auditing schemes. Both show that our CAB is capable of solve the mutual trust problem between DOs and CSPs in cloud storage system practically.

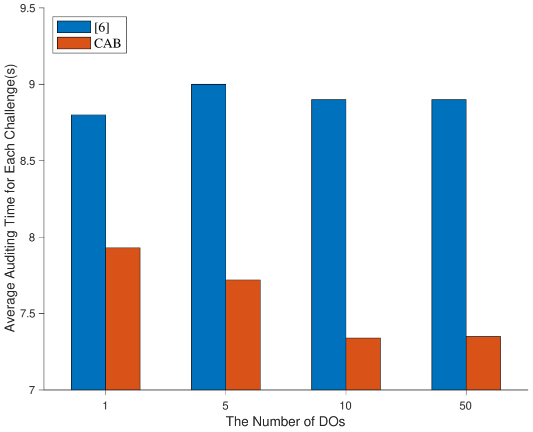

FIGURE 9. The average auditing time comparison for multiple DOs in a group in batch auditing.

REFERENCES

[1] P. Mell, T. Grance et al., “The nist definition of cloud computing,” 2011.
[2] K. Yang and X. Jia, “Data storage auditing service in cloud computing: challenges, methods and opportunities,” World Wide Web, vol. 15, no. 4, pp. 409–428, 2012.
[3] D. A. Fernandes, L. F. Soares, J. V. Gomes, M. M. Freire, and P. R. Inácio, “Security issues in cloud environments: a survey,” International Journal of Information Security, vol. 13, no. 2, pp. 113–170, 2014.
[4] L. F. Soares, D. A. Fernandes, J. V. Gomes, M. M. Freire, and P. R. Inácio, “Cloud security: state of the art,” in Security, Privacy and Trust in Cloud Systems. Springer, 2014, pp. 3–44.
[5] S. Nakamoto et al., “Bitcoin: A peer-to-peer electronic cash system,” 2008.
[6] H. Yu, Z. Yang, and R. O. Sinnott, “Decentralized big data auditing for smart city environments leveraging blockchain technology,” IEEE Access, vol. 7, pp. 6288–6296, 2018.
[7] J. Xue, C. Xu, J. Zhao, and J. Ma, “Identity-based public auditing for cloud storage systems against malicious auditors via blockchain,” Science China Information Sciences, vol. 62, no. 3, p. 32104, 2019.
[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in Proceedings of the 14th ACM conference on Computer and communications security. Acm, 2007, pp. 598–609.
[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling public verifiability and data dynamics for storage security in cloud computing,” in European symposium on research in computer security. Springer, 2009, pp. 355–370.
[10] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, “Blockchain based data integrity service framework for iot data,” in 2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017, pp. 468–475.
[11] C. Yang, X. Chen, and Y. Xiang, “Blockchain-based publicly verifiable data deletion scheme for cloud storage,” Journal of Network and Computer Applications, vol. 103, pp. 185–193, 2018.

[12] Y. QI and Y. HUANG, "Dira: Enabling decentralized data integrity and reputation audit via blockchain," SCIENCE CHINA Technological Sciences, vol. 62, no. 4, pp. 698–701, 2018.

[13] C. Li, J. Hu, K. Zhou, Y. Wang, and H. Deng, "Using blockchain for data auditing in cloud storage," in International Conference on Cloud Computing and Security. Springer, 2018, pp. 335–345.

[14] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication netowrks. ACM, 2008, p. 9.

[15] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," IEEE transactions on parallel and distributed systems, vol. 22, no. 5, pp. 847–859, 2010.

[16] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in Proceedings of the 2011 ACM Symposium on Applied Computing, 2011, pp. 1550–1557.

[17] 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 Transactions on Services Computing, vol. 10, no. 5, pp. 701–714, 2015.

[18] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper, August, vol. 19, 2012.

[19] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov, "A provably secure proof-of-stake blockchain protocol." IACR Cryptology ePrint Archive, vol. 2016, p. 889, 2016.

[20] M. Castro, B. Liskov et al., "Practical byzantine fault tolerance," in OSDI, vol. 99, no. 1999, 1999, pp. 173–186.

[21] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in p2p cloud storage," in 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2018, pp. 561–568.

PEI HUANG received his B. E. degree in Computer Science and Technology from Harbin Institute of Technology in 2018. Now he is a master student at State Key Laboratory of Integrated Service Networks of Xidian University, Xi'an, China, studying for his M. S. degree in Cyber Security. His research interests are IoT security, blockchain, and cloud storage security. His email address is galahad_moye@163.com.

KAI FAN is a professor at State Key Laboratory of Integrated Service Networks of Xidian University, Xi'an, China. In 2002, 2005, and 2007, he received his B. S. degree in telecommunications engineering, M. S. degree in cryptography, and Ph. D. degree in telecommunications and information system from Xidian University respectively. His research interests include cloud security, IoT security, network and information security. His email address is kfan@mail.xidian.edu.cn.

HANZHE YANG is a master student at State Key Laboratory of Integrated Service Networks of Xidian University, Xi'an, China. He received his B.E. degree in Software Engineering from Tiangong University at Tianjin in 2019. Now he is studying for his M.S. degree in Cyber Security from Xidian University. His research interests are blockchain, cloud storage security. His email address is 654602672@qq.com.

KUAN ZHANG is working as an assistant professor in Department of Electrical and Computer Engineering at University of Nebraska–Lincoln, USA. He received his B.S. and M.S. degrees from Northeastern University, P. R. China, in 2009 and 2011, respectively, in Communication Engineering and Computer Applied Technology. He received his Ph.D. degree from University of Waterloo, Canada, in 2016, in Electrical and Computer Engineering. He was a Postdoctoral Fellow from 2016-2017 at the University of Waterloo, Canada. He has published over 50 papers in journals and conferences. He was the recipient of Best Paper Award in IEEE WCNC 2013 and Securecomm 2016. His research interests include cyber security, big data, cloud/edge computing. The mailing address is PKI 206D, Scott Campus (Omaha), University of Nebraska–Lincoln, 1400 R Street Lincoln, NE 68588, USA. His email address is kuan.zhang@unl.edu.

HUI LI is a professor at State Key Laboratory of Integrated Service Networks of Xidian University, Xi'an, China. In 1990, he received his B. S. degree in radio electronics from Fudan University. In 1993, and 1998, he received his M. S. degree and Ph. D. degree in telecommunications and information system from Xidian University respectively. His research interests include network and information security. His email address is lihui@mail.xidian.edu.cn.

YINTANG YANG is a professor at Key Laboratory of Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices of Xidian University, Xi'an China. He received his Ph. D. degree in semiconductor from Xidian University. His research interests include semiconductor materials and devices, network and information security. His email address is ytyang@xidian.edu.cn.

• • •