Hindawi
Security and Communication Networks
Volume 2021, Article ID 9993855, 11 pages
https://doi.org/10.1155/2021/9993855

Research Article

PMAB: A Public Mutual Audit Blockchain for Outsourced Data in Cloud Storage

Hanzhe Yang[1], **Ruidan Su**[1], **Pei Huang**[1], **Yuhan Bai**[1], **Kai Fan**[1], **Kan Yang**[2], **Hui Li**[1] and **Yintang Yang**[3]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China*
[2]*Department of Computer Science, University of Memphis, Memphis 38152, TN, USA*
[3]*Key Laboratory of the Ministry of Education for Wide BandGap Semiconductor Materials and Devices, Xidian University, Xi'an 710071, China*

Correspondence should be addressed to Ruidan Su; rdsu@xidian.edu.cn

Received 4 March 2021; Accepted 19 May 2021; Published 2 June 2021

Academic Editor: Qi Li

With the rapid growth of data, limited by the storage capacity, more and more IoT applications choose to outsource data to Cloud Service Providers (CSPs). But, in such scenarios, outsourced data in cloud storage can be easily corrupted and difficult to be found in time, which brings about potential security issues. Thus, Provable Data Possession (PDP) protocol has been extensively researched due to its capability of supporting efficient audit for outsourced data in cloud. However, most PDP schemes require the Third-Party Auditor (TPA) to audit data for Data Owners (DOs), which requires the TPA to be trustworthy and fair. To eliminate the TPA, we present a Public Mutual Audit Blockchain (PMAB) for outsourced data in cloud storage. We first propose an audit chain architecture based on Ouroboros and an incentive mechanism based on credit to allow CSPs to audit each other mutually with anticollusion (any CSP is not willing to help other CSPs conceal data problems). Then, we design an audit protocol to achieve public audit efficiently with low cost of audit verification. Rigorous analysis explains the security of PMAB using game theory, and performance analysis shows the efficiency of PMAB using the real-world dataset.

1. Introduction

With the rapid technological advancements in Internet of Things (IoT), more terminals and better transmission efficiency also mean that mass data is generated while providing more convenience [1]. Massive terminal data and limited storage capacity make these IoT applications have to turn to Cloud Service Providers (CSPs) to obtain professional data storage support as Data Owners (DOs). In other words, technological advancements promote the integration of cloud services and IoT. In particular, cloud services are located in the data layer of IoT and interact with application servers to provide data services [2].

However, cloud services not only provide convenience for IoT but also challenge the privacy and security of data generated by terminals [3, 4]. As the data is stored in the cloud, the Data Owner will lose the strong control over the data. CSPs may be damaged by external threats, such as hacking or natural disasters, and even they may tamper with data for their own benefit. These external and internal attacks can damage the integrity of remote data [5]. If the integrity of data cannot be audited in time, with the damaged data being used for key calculation or operation, incalculable disaster will be triggered. The remote outsourcing data audit technology can assure the data integrity with only a small amount of interaction, which can just solve the above-mentioned security problems.

In 2008, Ateniese et al. [6] first proposed a partially dynamic Provable Data Possession (PDP) protocol. As a classic remote outsourcing data audit technology, PDP later developed the characteristics of dynamic audit, batch verification, and public audit [7–11]. The traditional public audit involves the interaction between multiple parties, which leads to the trust problem. For example, centralized storage makes audit results easy to be tampered with, TPA may help CSPs conceal data problems for profit, and so on.

The problem of multiparty trust in traditional data integrity audit makes it an inevitable trend to integrate blockchain technology into data integrity audit [12]. Yue et al. [12] and Liu et al. [13], respectively, proposed the prototype of data integrity audit framework combining IoT and P2P cloud storage environment with blockchain, but its application scenarios are relatively limited. Yu et al. [14] used blockchain for audit proof storage, and the Data Owner completed the audit of data integrity by verifying the audit proof stored on blockchain. Xu et al. [15] used blockchain to arbitrate disputed audit results. Huang et al. [16] completed verification of audit tasks and record of dynamic operations through representative nodes of the consortium chain built by PBFT consensus. Lu et al. [17] used Fabric (Consortium Blockchain) to store audit records and proposed a reputation system for TPA. TPA is an entity that makes profit through audit. The remuneration paid by DOs to TPA must be less than the actual value of the audited data; otherwise, the audit will be meaningless. Therefore, TPA is easy to be bribed by the benefits (more than audit remuneration but less than data value) paid by malicious CSP. In this case, collusion attacks are difficult to avoid.

Fan et al. [18] proposed an automated audit architecture based on Ethereum (Common Blockchain), which uses smart contracts to perform audit tasks and pay related compensation. Although Common Blockchain can effectively avoid collusion attacks because of its large scale of consensus nodes and effective incentive mechanism, it is difficult to reach an acceptable execution efficiency under larger-scale audit verification. Despite the fact that Consortium Blockchain is more efficient, there still exists the nothing-at-the-attack [19]. Without an effective incentive mechanism, collusion attacks will not be well resisted. PMAB is based on Consortium Blockchain and ensures mutual supervision through effective credit-based incentive mechanism, which strengthens the supervision of CSPs while auditing data. In [18], Verifiable Delay Function (VDF) is used to realize automatic audit; that is, the system automatically generates secure random source to generate audit challenge without DOs' participation, which further reduces the cost of DOs. However, the security of the random source comes from the continuous computing power consumption, which is not efficient enough. Therefore, due to the lack of customized blockchain design for audit protocol, the existing schemes still suffer from excessive overheads and collusion attacks.

To tackle the above challenges, we propose a Public Mutual Auditing Blockchain (PMAB) for outsourced data in cloud storage to solve collusion attacks in the public audit scheme, greatly reduce the audit cost, and improve the audit efficiency. The contributions of this paper can be summarized as follows:

(i) We present a customized blockchain architecture PMAB for public audit, which enables all CSPs to automatically audit each other through audit contract and releases DOs from data audit cost

(ii) We propose a credit-based incentive mechanism to resist collusion attacks while quantifying behaviors of entities

(iii) We put forward a consensus for PMAB that combines an efficient public audit protocol. After rigorous security and performance analysis, our scheme can achieve expected security goal and audit efficiency significantly ahead of existing schemes

The outline of this paper is as follows: we first introduce the background knowledge, the system model, threat model, and design goals. In the latter, we describe the concrete constructions of PMAB and audit protocol. After that, security and performance analyses are detailed. Finally, the summary and future work of this paper are presented.

2. Preliminaries

2.1. Ouroboros. Ouroboros is a kind of blockchain consensus based on Proof of Stake (PoS), which was proposed by Kiayias et al. [20] and proved secure. It uses Publicly Verifiable Secret Sharing Scheme (PVSS) [21] to generate antibiased random numbers as random source of the representative election algorithm Follow the Satoshi (FTS), so that the candidate can be elected as the representative node with a certain probability, which is equal to the proportion of the candidate's stake to the overall stake of all candidates.

3. Problem Statement

3.1. System Model. PMAB considers a public data audit scenario for outsourced data in cloud storage, which is mainly composed of Data Owner (DO), Cloud Service Providers (CSPs), and Regulator (R) as shown in Figure 1. Audit chain and credit chain are two distributed ledgers maintained by CSPs and *R*, which, respectively, record audit information and credit of each entity. After outsourcing data to CSPs, DO (e.g., an IoT application collects data via their terminals) generates the audit contract with CSPs and *R* (Steps 1 and 2). In public audit, the audited CSP provides proof to the audit chain according to the challenge (Steps 3 and 4); then some CSPs complete audit verification and credit settlement under the supervision of *R* (Step 5). Finally, DO can obtain audit and credit settlement results through these two distributed ledgers (Step 6). The specific roles of all entities in PMAB are described as follows:

(i) **DO** has limited communication, computation, and storage resources. It outsources data to CSPs and achieves public audit with PMAB

(ii) **CSP** provides DOs with significant storage space and computation capability. It is also responsible for maintaining two distributed ledgers, while responding proof to challenges and completing public audit

(iii) **R** is also responsible for maintaining two distributed ledgers while supervising public audit process and administrating PMAB

3.2. Threat Model. PMAB considers that some corrupted CSPs will try to bribe other CSPs to conceal their data problem in audit verification. DO is honest but curious; it

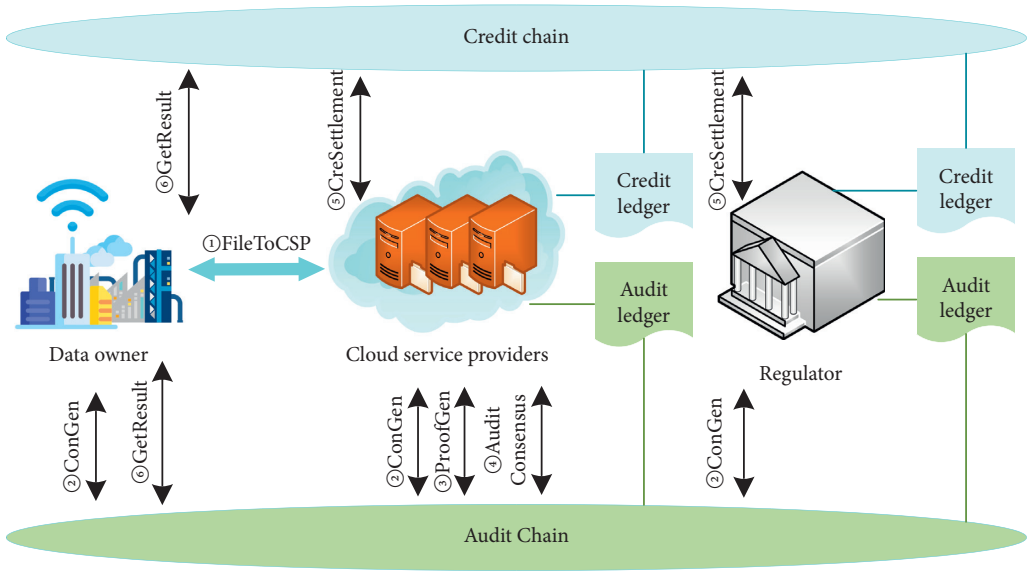


Figure 1: System model of PMAB.

will try to obtain the identity and audited outsourcing data of other DOs based on the audit information from audit chain. R is assumed to be a trustworthy regulatory agency that supervises cloud storage services.

3.3. Design Goals. To achieve secure and efficient automated data audit under the above threat model, PMAB should achieve the following goals about anticollusion, privacy preserving, efficiency, automated audit, and dynamic audit:

Anticollusion. PMAB should prevent corrupted CSPs from passing audit verification through collusion attacks

Privacy Preserving. Except for R, CSP, and DO participating in the audit contract, all other entities cannot obtain the specific identity and outsourced data information of the DO

Antiforgery. The audit proof forged by malicious CSP cannot pass the audit verification

Antireplace. For malicious CSP, when generating audit proof, it cannot use the combination of intact data block related information to get the proof of damaged data block

Efficiency. The average cost of batch audit in the audit protocol of PMAB should be limited to a very low and constant level, and the overall verification and the consensus time of PMAB should be controlled within a limited time

Automated Audit. PMAB should achieve automatic audit periodically based on audit contracts

Dynamic Audit. The remote data that is modified dynamically can be audited timely and effectively

4. Public Mutual Audit Blockchain

4.1. Design Overview of PMAB. As the analysis above, all public audit schemes based on blockchain cannot audit efficiently and resist collusion attacks at the same time.

In PMAB, we innovatively use the mutual audit between CSPs instead of TPA's audit. According to the game theory, we design an incentive mechanism based on credit, so that no CSP is willing to help other CSPs conceal data problems. Furthermore, based on Ouroboros [20], we design an audit protocol that combines with the blockchain consensus to efficiently and automatically complete public audits.

Therefore, the description of PMAB is mainly divided into two parts: basic blockchain structure and audit protocol.

4.2. Basic Blockchain Structure of PMAB. Blockchain architecture is the basic design of PMAB, which is mainly composed of two parts, namely, credit chain and audit chain. In this part, we will introduce the core of credit chain (i.e., the incentive mechanism) and the basic data structure in audit chain.

4.2.1. Incentive Mechanism. Incentive mechanism is the power source and security cornerstone of blockchain system. The credit value credit is the core of PMAB incentive mechanism, which mainly comes from CSPs' initial Credit, deposit, and audit remuneration reward paid by DOs. The candidate node with higher credit is more likely to be elected as a representative node. Moreover, the credit lost by collusion will outweigh the creditgained, and rational CSPs will conduct honest audits to maximize benefits. Some key concepts related to creditare described below:

(i) *initial Credit*. When each CSP joins PMAB, it needs to pay some deposits in exchange for *initial Credit*, which will be confiscated when the malicious behavior of this CSP is found. Only when initial Credit reaches the threshold can it become a candidate node.

(ii) *deposit*. When the audit contract is constructed, the CSP needs to mortgage *deposit*, of which *dataValue* and *penalty* are equal in half.

(iii) *dataValue*. As the compensation that CSP pays to DO when audit fails, it represents the value of data.

(iv) *penalty*. As a fine for the malicious behavior of CSP when being audited.

(v) *bonusPool*. All forfeited *initialCredit* and penalty will be put into *bonusPool*, and the honest CSPs participating in the audit will divide up *bonusPool*.

4.2.2. Block and AuditContract. *Block* and *AuditContract* are basic data structure in audit chain. *Block* stores the contents and results of each audit. *AuditContract* keeps the specific information of each audit task. The whole contract is stored in *R*, the associated DO, and CSP, all CSPs just keep *conHeader*, which determines the audit task information of each consensus.

The structures of *Block* and *AuditContract* are shown in Figures 2(a) and 2(b). Descriptions of key fields are as follows:

(i) *nonce*. A random value obtained by PVSS [21] in Ouroboros [20] is used as random source for this audit

(ii) *proof*. All audit proofs collected by the representative node during the audit

(iii) *auditCons*. The collection of audit tasks covered in this audit

(iv) *verResult*. The results of this audit

(v) *ConPk*. The public key to be used in the audit verification

(vi) *auditRate*. The proportion of audited data to outsourced data

(vii) *Rproof*. A proof of the overall stored data provided by *R*

4.3. High Description of Audit Protocol. This part focuses on the audit protocol of PMAB, which is divided into Setup phase and Audit phase. There are system parameters initialization and audit preparation in Setup phase. Audit phase includes the generation and verification of audit proof, as well as credit settlement. In addition, in order to verify the remote data of dynamic operation in time, PMAB supports dynamic audit.

4.3.1. Setup Phase. In this phase, the system parameters are first initialized in *KeyGen*. Then CSPs join PMAB in *SystemIni*. After DO preprocesses files which will be outsourced and uploads them to CSP in FileToCSP, the *AuditContract* is constructed by DO, CSP, and *R* in *AuditConGen*.

KeyGen. With a security parameter λ, two elliptic curve groups G_1 and G_2 and a multiplicative group G_T of the large prime order p, a bilinear pairing e: $G_1 \times G_2 \longrightarrow G_T$, a field Z_p of residue classes modulo p, two random generators $g_1 \in G_1$ and $g_2 \in G_2$, a pseudorandom permutation (PRP) $\pi(\cdot)$, and a pseudorandom function (PRF) $f(\cdot)$ are picked.

$$S\mathbb{P} = \{G_1, G_2, G_T, g_1, g_2, e, \pi, f\}. \quad (1)$$

Furthermore, for the convenience of expression, $\sigma_\epsilon(\cdot)$ is used to represent a signature signed by entity ϵ and ID_ϵ is used to represent the unique identicator of entity ϵ.

SystemIni. After the new CSP exchanges *initialCredit* (Icr) from *R*, *R* broadcasts new node access notification NAN = $(t, ID_{CSP}, CSP_{\text{address}}, Icr, \sigma_R(\text{NAN}))$ to all CSP nodes, where t is the timestamp and CSP_{address} is the network address of new CSP. After receiving the NAN, other CSP nodes establish the connection with new node.

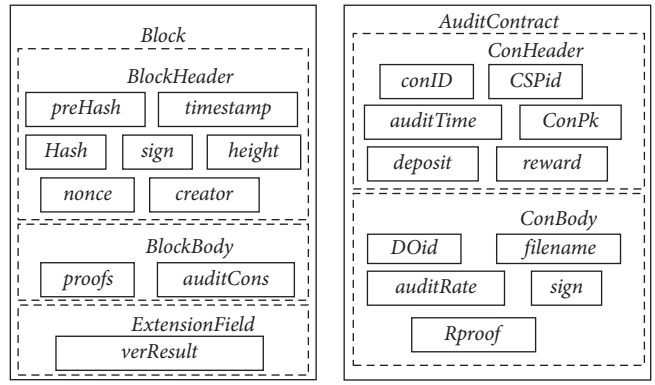
FileToCSP. Assuming that the file that DO needs to store is*F*, DO divides *F*into following data blocks:$F = \{m_1, m_2, \ldots, m_i, \ldots, m_n\}, i \in [1, n], m_i \in Z_p$. DO generates a random parameter $\omega_F \in Z_p^*$ for *F*, thereby obtaining the verification random number set $R_F = \{r_i\}, i \in [1, n]$, where $r_i = f_{\omega_F}(i)$. Then $\text{sk} = \alpha \in Z_p^*$ is randomly selected as audit private key; thereby BLS homomorphic verification tags $\sigma_i = (g_1^{(m_i + r_i)})^\alpha$ for each data block m_i are generated, thereby obtaining a tag set $\sigma = \{\sigma_i\}, i \in [1, n]$. Finally, DO sends a tag collection message $\text{TC} = \{t, F, \sigma, \sigma_{DO}(\text{TC})\}$ to the CSP that stores outsourced data.

AuditConGen. Assuming that DO and CSP have negotiated *deposit*, *auditRate*, *auditTime*, and other information for *AuditContract*, DO generates audit public key $ConPk = g_2^\alpha$ and sends R_F to R. After *R* generates$Rproof = g_1^{(r_1 + r_2 + \ldots + r_n)}$, CSP fills all information in the *AuditContract*, especially$sign = \sigma_{CSP|DO|R}(AuditContract)$. Then DO can delete *F* and σ locally. After receiving $con = (t, ID_R, ConHeader, \sigma_R(con))$, each CSP stores *ConHeader* in local contract collection Con.

4.3.2. Audit Phase. This part mainly focuses on the detailed process of data audit. Before each round of audit, the CSP whose *initialCredit* reaches the threshold will participate in representative election as a candidate. After PVSS and FTS [17, 18], we get random source *Random* and a representative *Rep*, and other candidates become endorsers Endo. After the audited CSP obtains the audit proof *P* through *ProofGen* based on *Random*, PMAB completes the audit consensus and appends *Block* to audit chain in *AuditConsensus*. After *CreSettlement*, audit result *verResult* is added to audit chain and *credit* is updated to credit chain. Finally, DO can obtain audit results related to itself from audit chain. For ease of understanding, the following description is based on a scenario, where a CSP is audited by multiple DOs.

ProofGen. After obtaining the random Source *Random*, each CSP checks whether it needs to be audited this *round based on local contract collection Con*. If it does, the corresponding challenge set *chal* will be calculated according to *Random*, and the audit proof set *P* will be generated.

CSP first generates two keys $k_1 = f_{\text{Random}}(\text{height})$ and$k_2 = f_{\text{Random}}(\text{height} + 1)$. The audit contract set



(a) (b)

Figure 2: The structure of *Block* and *AuditContract*. (a) The structure of *Block*. (b) The structure of *AuditContract*.

$ConCache = \{Contract_j\}_{j\in[1,K]}$ that needs to be executed by the CSP in this round is generated, where K represents the number of audit contracts in *ConCache*. According to $auditRate_j$ of $Contract_j$ in *ConCache* and the actual size n_j of the audited file, the number of challenged blocks for this audit contract z_j is computed as $z_j = \lceil auditRate_j \times n_j \rceil$. Furthermore, the challenge set of current round $Chal = \{chal_j\}_{j\in[1,K]}$ of the CSP is obtained, where $chal_j = \{i_l, v_l\}, i_l = \pi_{k_1}(l), v_l = f_{k_2}(l), l \in [1, z_j]$. CSP then calculates the tag proof $TP_j = \prod_{chal_j} \sigma_{i_l}^{v_l}$ and the data block proof $DP_j = g_1^{\sum_{chal_j} m_{i_l}\cdot v_l}$ corresponding to each $chal_j$, thereby forming the tag proof set $\Phi = \{\prod_{\text{Chal}} \text{TP}_j\}$ and the data block proof set $\mu = \{DP_j\}_{j\in[1,K]}$. Finally, the proof set $P = \{\Phi, \mu\}$ of the CSP is obtained.

AuditConsensus. As shown in Figure 3 an audit consensus is conducted after *ProofGen*. First, the audited CSP sends its own proof message $proof = \{t, P, \sigma_{CSP}(proof)\}$ to *Rep* and Endo nodes. *Rep* packs received *proof* into a message $proofs = \{t, \{P\}, \sigma_{\text{Re}p}(proofs)\}$ and broadcasts it to all *Endo* nodes, where N represents the number of nodes that need to execute audit contracts. After receiving*proofs*, Endo nodes compare it to *proof* they received; if *proof* is included in *proofs*, they will pack the message $response = \{t, proofs, \sigma_{En\ do}(response)\}$ and send it to *Rep*. After collecting allresponse, *Rep* packs the message $Rproof\ \text{Re}quest = \{t, \{response_s\}_{s\in[1,N]}, \sigma_{CSP_{\text{Re}p}}(Rproof\ \text{Re}quest)\}$ and sends it to *R*. To verify $Rproof Request$, *R* compares *proofs* of all response in $Rproof Request$. If they are the same, *R* will calculate the random number proof $proof_R = \{\xi_s\}_{s\in[1,N]}$ required for this round of proof consensus, where $\xi_s = \left\{RP_j = g_1^{\sum_{chal_j} r_{i_l}\cdot v_l}\right\}_{j\in[1,K]}$, and then package message $Rproof Response = \{t, proof_R, \sigma_R(Rproof Response)\}$ and send it to *Rep*.

Then, *Rep* fills $Rproof Request|Rproof Response$ in *proofs*, *ConCache* in *auditCons*, *Random* in nonce, and so on while generating a new *Block*.

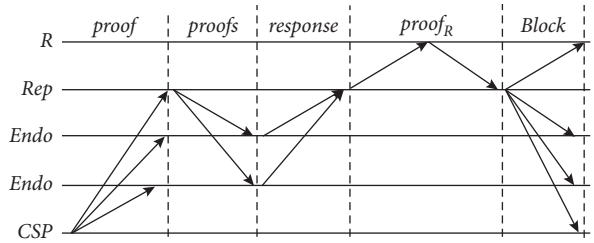


Figure 3: The process for generating new *Block*.

Finally, *Block* is broadcast to all CSPs and *R*.

CreSettlement. After *AuditConsensus*, all *Endo* and *Rep* nodes verify the proof P to audit the outsourced data. The verification operation is to verify whether the following equation holds for audited CSP.

$$e(\Phi, g_2) = \prod_{ConCache} e\left(DP_j \cdot RP_j, ConPk_j\right). \quad (2)$$

If it holds, $Ver = true$; otherwise, $Ver = false$. Then message $verify = \{t, Ver, \sigma_{Rep/En\ do}(verify)\}$ is sent to *R*.

After receiving all $verify$, *R* verifies whether there are different verification results. If all $verify$ are the same, verification result set *RVer* and the malicious nodes set *Mal* will be empty (i.e, all *Endo* and *Rep* are honest); otherwise, *R* will use equation (2) to further verify proofs for the dispute and then get $RVer = \{verify_s\}_{s\in[1,A]}$ and $Mal = \{ID_{Endo_s/Rep_s}\}_{s\in[1,M]}$, where A represents the number of disputed proofs and Mrepresents the number of malicious nodes. After receiving $ack = \{\{Ver_s\}_{s\in[1,N]}, RVer, Mal, t, \sigma_R(ack)\}$ from *R*, all nodes put it into *verResult* of the corresponding *Block* in audit chain. Then all CSPs and *R* will conduct credit settlement based onack. First, the total reward *totalReward* of all executed audit contracts in *ConCache* is calculated and put in *bonusPool*, and the DO in the audit contract is compensated for data corruption. If *dataValue*, *Mal* is not empty, all *initialCredits* of the CSP and *penalty* in the corresponding audit contract involved are put into *bonusPool*. Finally, *credit* in *bonusPool* will be

obtained by virtuous *Endo* and *Rep* (*Rep* can get an extra part).

To prove the correctness of audit process, equation (2) can be derived as follows:

$$\begin{aligned} e(\Phi, g_2) &= e\left(\prod_{Chal}\prod_{Chal_J}\left(\left(g_1^{\left(m_{i_l}+r_{i_l}\right)}\right)^{\alpha_j}\right)^{v_l}, g_2\right) \\ &= \prod_{ConCache} e\left(\prod_{chal_j}\left(g_1^{\left(m_{i_l}+r_{i_l}\right)}\right)^{v_l}, g_2^{\alpha_j}\right) \\ &= \prod_{ConCache} e\left(g^{\sum_{chal_j} m_{i_l}\cdot v_l + \sum_{chal_j} r_{i_l}\cdot v_l}, ConPk_j\right) \\ &= \prod_{ConCache} e\left(DP_j \cdot RP_j, ConPk_j\right). \end{aligned} \tag{3}$$

4.3.3. Dynamic Audit. EMAB supports dynamic auditing; that is, it supports DO in auditing data after dynamical operations, which mainly consist of insertion, modification, and deletion. The details are described below.

(i) *Insertion*. Suppose that DO wants to insert a data block m_j in file F, j is the index position to be inserted, and $1 \le j \le n+1$, where $n$represents the number of data blocks of origin F. DO updates the local FIT data (the auxiliary data linked list corresponding to file F) and inserts the new node $(B_j = n+1, r_j = f_{\omega_F}(n+1))$ into the $j-$th position of FIT. DO calculates the BLS-HVT $\sigma_j = (g_1^{(m_j+r_j)})^{\alpha}$ of m_j and sends the message $insert = \{t, j, m_j, \sigma_j, \sigma_{DO}(insert)\}$ to the CSP to help update m_j and σ_j. The message $up\ da\ teRproof = \{t, j, r_j, \sigma_{DO}(up\ da\ te\ Rproof)\}$ is sent to R to help update R_F.

(ii) *Modification*. Suppose that DO wants to update the data block m_j in file F. The message $up\ da\ te = \{t, j, m_j, \sigma_j, \sigma_{DO}(update)\}$ is sent to the CSP to help it update the data block m_j and BLS-HVT σ_j, where $\sigma_j = (g_1^{m_j+r_j})^{\alpha}$.

(iii) *Deletion*. Suppose that DO wants to delete data block m_j in file F. DO moves the $j-$th node in F's FIT to the end of the chain and sets its B_j to -1. The message $de\ lete = \{t, j, \sigma_{DO}(delete)\}$ is sent to CSP to help it delete the corresponding data block m_j and BLS-HVT σ_j. The message $de\ leteRproof = \{t, j, \sigma_{DO}(deleteRproof)\}$ is sent to R to help it delete the corresponding random number element r_j.

All dynamic operation records are stored in the dynamic operation domain of corresponding audit contract after all participants sign. In the following audit consensus, all new data will be applied.

5. Security Analysis

In this part, we mainly analyze anticollusion, privacy preserving, antiforgery, and antireplace described in Section 3.3.

5.1. Anticollusion. PMAB can resist collusion attacks from consensus nodes (*Rep* and *Endo*). Colluding nodes cannot deceive R by sending wrong verify to bypass corrupted data blocks. They definitely betray each other because honest behavior is more profitable than collusion.

Because consensus nodes will send $verify$ to R at the same time in each round of audit consensus, this process can be regarded as a static and complete information game. For simplicity, we take two consensus nodes, namely, $player_1$ and $player_2$, for example. Suppose that v_1 and v_2 are $dataValue$ of $player_1$ and $player_2$, respectively, p_1 and p_2 are *initialCredit* (and $penalty$) of $player_1$ and $player_2$ $(v_1 < p_1, v_1 < p_2, v_2 < p_1, v_2 < p_1)$, m, $(0 < m < v_2)$ is the cost of bribery, and u is the reward of audit.

The game elements are as follows:

(i) $players\{player_1, player_2\}$

(ii) $strategy\{honest, malicious\}$

(iii) $utility$ Profit matrix when both $players$ have data problems and only $player_2$ has data problems as Tables 1 and 2 show, respectively

When both players have data problems, for $player_2$, it is easy to know $u - v_2 > u - p_2 - v_2$ and $p_1 + u - v_2 > u$, so honest must be the dominant strategy of $player_2$. Similarly, it is easy to know that for, $player_1$, *honest* is also the dominant strategy. So, the Nash equilibrium point in this case falls in case (*honest*, *honest*). Therefore, in this case, no collusion problem occurs. When only $player_2$ has data problems, for $player_2$, it is easy to know $u - v_2 > u - p_2 - v_2$ and $p_1 + u - v_2 > u - m$, so *honest* must be the dominant strategy of $player_2$. For $player_1$, it is easy to know $u > u - p_1$ and $P_2 + u > u + m$, so *honest* must be the dominant strategy of $player_1$. So the Nash equilibrium point in this case falls in case (*honest*, *honest*). Therefore, in this case, no collusion problem occurs.

The situation of more players is similar to the situation of two players. In summary, PMAB can avoid collusion problems.

5.2. Privacy Preserving. Apart from R and audited CSP, all other CSPs cannot obtain the relationship between audit tasks and DOs from *Con*, and the specific data block information from P, that is, PMAB, can protect DO's identity privacy and data privacy.

Identity Privacy Protection. All *ConHeader* are stored in CSPs' local contract collection *Con*. Only the audit public key *ConPk* in *ConHeader* is associated with DO, but *ConPk* of each *ConHeader* of DO can be different. If there is no duplicate *ConPk*, it is impossible to get the association between *ConPk* and DO. So the privacy of the DO's identity is protected.

Table 1: Profit matrix when both *players* have data problems.

		$player_2$	
		honest	*malicious*
$player_1$	*honest*	$u - v_1, u - v_2$	$p_2 + u - v_1, u - p_2 - v_2$
	malicious	$u - p_1 - v_1, p_1 + u - v_2$	u, u

Table 2: Profit matrix when only $player_2$ has data problems.

		$player_2$	
		honest	*malicious*
$player_1$	*honest*	$u, u - v_2,$	$p_2 + u, u - p_2 - v_2$
	malicious	$u - p_1, p_1 + u - v_2$	$u + m, u - m$

Data Privacy Protection. In the audit consensus, it is difficult for the consensus nodes to obtain $\sum_{chal_j} m_{i_l} \cdot v_l$ from $DP_j = g_1^{\sum_{chal_j} m_{i_l} \cdot v_l}$ because of DLP. Moreover, even if $\sum_{chal_j} m_{i_l} \cdot v_l$ is given, the specific information of m_{i_l} cannot be solved out without knowing the number of $m_{i_l} \cdot v_l$. Therefore, PMAB can ensure that consensus nodes cannot obtain the data information of the audit data during the verification process, which protects data privacy.

5.3. Antiforgery. If the data block m_i within *chal* has been modified to $m_i + off_i$ by the CSP, where off_i denotes the modification part, to adapt the new DP^*, a new TP^* should be computed as follows:

$$\begin{aligned} TP^* &= \prod_{chal} \left(\left(g_1^{m_{i_l} + r_{i_l} + off_{i_l}} \right)^{sk} \right)^{v_l} \\ &= \prod_{chal} \left(\left(g_1^{m_{i_l} + r_{i_l}} \right)^{sk} \cdot \left(g_1^{off_{i_l}} \right)^{sk} \right)^{v_l} \\ &= \prod_{chal} \sigma_{i_l}^{v_l} \cdot g_1^{sk \cdot \sum_{chal} off_{i_l} \cdot v_l} \\ &= TP \cdot g_1^{sk \cdot \sum_{chal} off_{i_l} \cdot v_l}. \end{aligned} \tag{4}$$

Because this CSP only owns TP, it needs to know sk for obtaining TP^*. However, *sk* is a private key of the DO. In our assumption, *sk* cannot be obtained by others. Hence, the audit proof cannot be forged by a CSP, and PMAB can resist forgery attack.

5.4. Antireplace. Suppose that a corrupted data block m_j has been checked, and two data blocks m_{j_1} and m_{j_2} are intact. To obtain the HVT of m_j, the correct combination of σ_{j_1} and σ_{j_2} should be found. Since $\sigma_{j_1} = (g_1^{(m_{j_1} + r_{j_1})a})^{sk}$, $\sigma_{j_2} = (g_1^{(m_{j_2} + r_{j_2})})^{sk}$, a CSP sets that

$$\begin{aligned} \sigma_j^* &= \sigma_{j_1}^{\alpha_{j_1}} \cdot \sigma_{j_2}^{\alpha_{j_2}} \\ &= \left(\left(g_1^{\left(m_{j_1} + r_{j_1} \right)} \right)^{sk} \right)^{\alpha_{j_1}} \cdot \left(\left(g_1^{\left(m_{j_2} + r_{j_2} \right)} \right)^{sk} \right)^{\alpha_{j_2}} \\ &= \left(g_1^{\alpha_{j_1} \cdot \left(m_{j_1} + r_{j_1} \right) + \alpha_{j_2} \cdot \left(m_{j_2} + r_{j_2} \right)} \right)^{sk}, \end{aligned} \tag{5}$$

where $\alpha_{j_1}, \alpha_{j_2} \in Z_p$. If σ_j^* is to be equal to σ_j, $\alpha_{j_1} \cdot (m_{j_1} + r_{j_1}) + \alpha_{j_2} \cdot (m_{j_2} + r_{j_2}) = m_j + r_j$ must be satisfied. In order to meet this requirement, r_{j_1}, r_{j_2}, and r_j must be known. But CSP cannot get them based on the information it already has. For example, if g_1, m_j, and $(g_1^{(m_j + r_j)})^{sk}$ are known, it is required to solve r_j. r_j is unknown, so $r_j + m_j$ is also unknown. If g_1 is given, solving $r_j + m_j$ from $(g_1^{(m_j + r_j)})^{sk}$ is a DLP problem. So in polynomial time the probability of solving r_j is negligible.

Similarly, solving r_{j_1} and r_{j_2} is the same as solving r_j. So replace attack from CSP can be resisted in EMAB.

6. Performance Analysis

In this part, we focus on the theoretical and experimental analyses of PMAB's performance through comparing them with similar schemes: Dredas [18], Fabric [17], and CAB [16]. The notations used in the performance analysis are shown in Table 3.

6.1. Theoretical Analysis. The comparison of entities' computation cost with Dredas [18], Fabric [17], and CAB [16], also supporting public audit by blockchain, is shown in Table 4.

Computation overheads are mainly distributed in *FileToCSP*, *ProofGen*, and *AuditConsensus* in the comparison. In order to provide the reference for comparison, we test 1000 times and then obtain the average cost of each operation; that is, $H = 18.98\,ms, E = 9.64\,ms, P = 4.63\,ms$, and $M = 2.51\,ms$. From Table 4, we can see that the DO and consensus node in the PMAB cost much less. Firstly, in *FileToCSP*, DO generates tags for all data blocks to be uploaded. In this part, compared with all other schemes, $nH + nM$ operations are avoided in PMAB. Then, in *ProofGen*, audited CSP computes challenges and corresponding proofs. In this part, $zM - E$ operations are avoided in PMAB, compared with other fastest schemes. Finally, in *AuditConsensus*, the smart contract in Dredas [18], the TPA in Fabric [17], or each consensus node in CAB [16] and PMAB verifies the correctness of proofs. Proof verification is the core part of public audit, and it is also the efficiency bottleneck of the whole public audit. In this part, the verification cost of Fabric and CAB increases linearly, while the verification cost of Dredas [18] and PMAB remains at a

Table 3: Notation definitions of performance analysis.

Notation	Description
H	Hash function mapping a string to a point on G_1 and G_2.
E	Modular exponentiation on G_1 and G_2.
P	Bilinear pairing operation of e.
M	Point multiplication on G_1 and G_2.
n	The total number of data blocks outsourced.
z	The number of challenged data blocks.

Table 4: Comparison of computation cost.

Schemes	DO	CSP	TPA	Consensus node
Dredas	$nH + 2nE + nM$	$zH + (2z + 1)E + 2(z - 1)M$	—	$2E + 2P + 2M$
Fabric	$nH + 2nE + nM$	$zE + (2z - 1)M$	$2P + (z + 1)E + zM$	—
CAB	$nH + 2nE + nM$	$zE + (2z - 1)M$	—	$zH + (z + 1)E + 3P + zM$
PMAB	$2nE$	$(z + 1)E + (z - 1)M$	—	$2P + M$

lower and constant level, and furthermore PMAB avoids $2E + M$ operations compared with Dredas [18].

6.2. Experimental Analysis. We evaluate performance of PMAB by conducting several experiments using JDK 1.8 on Ubuntu 16.04 system equipped with Intel Core i5-8400 CPU at 2.3 GHz and 4 GB RAM. We also use Docker to virtualize different nodes. WebSocket and Netty are used for TCP and HTTP communication, respectively. All pairing operations and related calculations on an elliptic curve 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. The signature algorithm is implemented by the identity-based signature in [22] with JPBC library. The hash algorithm implemented is SHA-512 in BouncyCastle library. The encryption and decryption algorithm uses RSA-1024 in the security library JCE (Java Cryptography Extension) of Java. The test datasets stem from China-Brazil Earth Resources Satellite (CBERS) on Amazon Web Service (AWS). The image files in CBERS are converted to Cloud Optimized GeoTIFF format in order to optimize its use for cloud-based applications. Each test file is divided into 10,000 4 KB data blocks. According to LFT (Loss Function Theory) presented in [12], the optimal balance between the high detection probability and the low verification cost can be achieved by challenging a limited number of data blocks. Therefore, the sample size of data blocks in our experiments is changed from 50 to 500.

FileToCSPTime. Figure 4 shows the computation cost of DO in *FileToCSP*. With the sample size increasing, it is obvious that the growth rate of *FileToCSP*'s computation cost in PMAB (0.91 s~8.85 s) is less than half as much as other schemes (1.85 s~18.21 s).

ProofGenTime. Figure 5 shows audited CSP's computation cost during generating audit proof. Dredas [18] takes significantly more time (1.81 s~18.04 s) because there are many heavy operations such as H and E on G in the proof

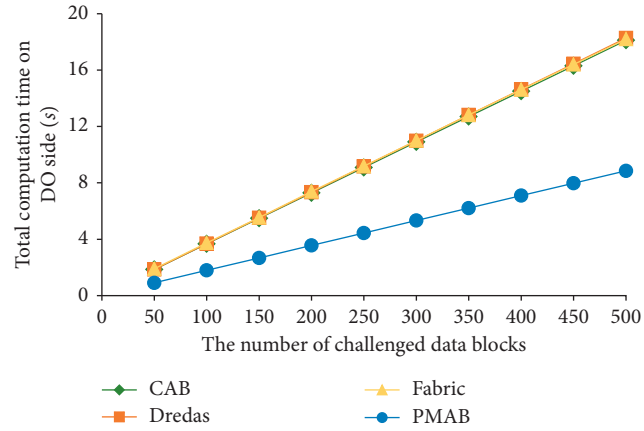


Figure 4: The computation cost comparison in *FileToCSP*.

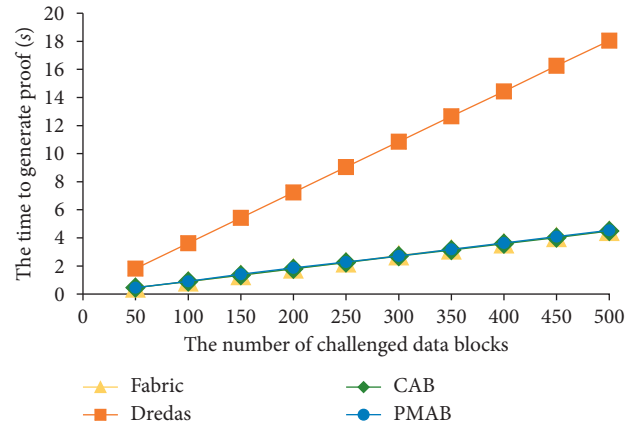


Figure 5: The computation cost comparison in *ProofGenTime*.

generation, while the proof generation times of PMAB and the remaining schemes are almost the same (0.45 s~4.54 s).

AuditVerifyTime. Figure 6 shows the verification time of the TPA in Fabric [17] and the consensus node in Dredas [18], CAB [16], and PMAB spend, respectively, where the

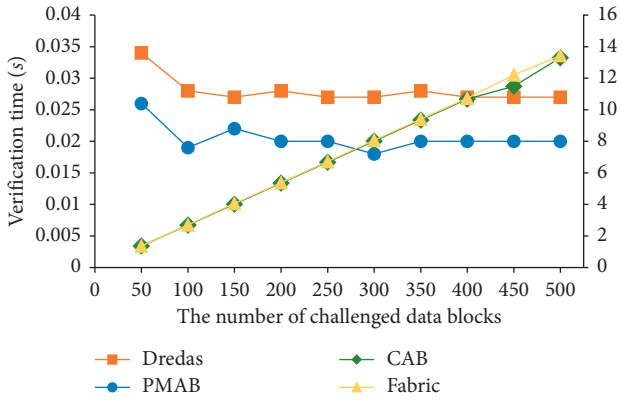


Figure 6: The verification cost comparison in *AuditVerifyTime*.

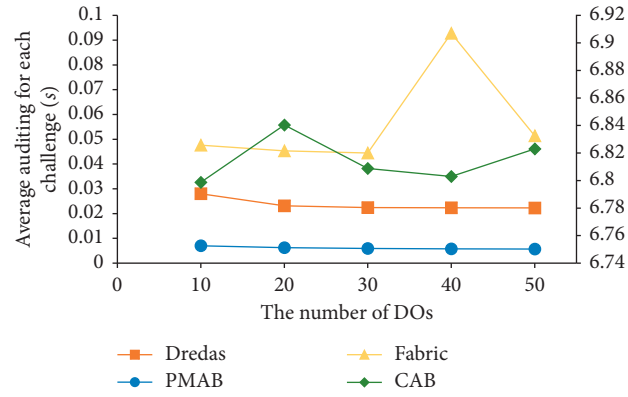


Figure 7: The computation cost comparison in *BatchAuditTime*.

verification time of Dredas [18] and PMAB ranges from 0.019 s to 0.034 s and the verification time of Fabric [17] and CAB [16] ranges from 1.36 s to 13.44 s. It is obvious that there is a linear relationship between the number of challenged blocks and the verification time of Fabric [17] and CAB [16], while the verification times of Dredas [18] and PMAB tend to be 27 ms and 20 ms, respectively. Thanks to less expensive operations such as H and E, our verification cost is more acceptable to each involved verifier compared with other schemes.

BatchAuditTime. In Figure 7, we compare the batch auditing with Dredas [18], Fabric [17], and CAB [16] under the condition that each DO challenges the same CSP with 250 data blocks in a challenge set. The average audit time of Dredas [18] and PMAB ranges from 0.007 s to 0.028 s, and the average audit time of Fabric [17] and CAB [16] ranges from 6.679 s to 6.83 s. It is obvious that, with the increase in the number of aggregated audit tasks, the average audit time cost of Fabric [17] and CAB [16] fluctuates between 6.8 s and 6.9 s, while Dredas [18] and PMAB tend to 0.02 s and 0.005 s, respectively. PMAB is four times faster than the fastest of other schemes in average audit time of the batch audit. Considering that the single validation time of PMAB in Figure 6 is only one-third faster than that of Dredas [18], our batch audit efficiency is higher.

RandomGenTime. In each consensus of the blockchain, a random source will be obtained to complete the current round of audit tasks, that is, automatic audit. The random sources in Dredas [18] are generated by a verifiable random function (VRF). In order to ensure the freshness and security of the generated random source, CSP needs to execute VRF by taking the nonce of the new Ethereum block as a seed, until the block is fully confirmed by the Ethereum network; that is, the block cannot be tampered with afterwards. Then the VRF is terminated, and the corresponding random source is obtained. The smart contract verifies the validity of the random source before using it. However, the validation process can be divided into K parallel tasks by using K process states provided by CSP, so the validation time is $(1/K)$ of the generation time. According to [23], the average time to generate a new block in Ethereum is 14 s. Generally, eight blocks are generated before the block is confirmed by the network, so it takes at least 112 s to get the random source. Assuming $K = 100$, that is, there are 100 parallel verification processes, the verification takes nearly 2 s. Therefore, the random source generation time of Dredas [18] is constant at 114 s. In the random source generation process of our scheme, each consensus node has to send and process $2(n-1)$ messages (n is the number of consensus nodes) and do n hash operations and $n-1$ encryption operations. If a node does not send open, each node should decrypt $E_i(open_i)$ it receives to solve this case. Figure 8 shows the comparison between PMAB and Dredas [18] in terms of the random source generation time, where the time cost of PMAB ranges from 0.25 s to 3 s and the time cost of Dredas [18] is always 114 s. The time cost of our random source generation method (PVSS [21]) increases linearly and slowly with increase in the number of consensus nodes. We roughly estimate that it will take more than 3000 consensus nodes to spend as much random source generation time as Dredas [18]. However, PMAB uses the *threshol d* of *initialCre di t* to limit the number of consensus nodes. Only a few consensus nodes are required to complete PVSS [21] and audit consensus. Therefore, PMAB's random source generation is more efficient.

ParallGenProof. In the face of large-scale audit case, according to the formation processes of Φ and μ, our protocol in PMAB actually supports parallel generation and aggregation of audit proofs employing MapReduce principle [24]. CSP will divide the whole task of proof generation into small tasks of the same scale for parallel execution and then aggregate the results of single tasks. We set 10 audit tasks as a group and make CSP process the audit proof generation process in parallel, testing the change of proof generation time when the number of audit tasks grows from 100 to 1000, in which 250 data blocks were questioned for each audit task. As shown in Figure 9, it is clear that when CSP is faced with a large number of requests, it proves that the generation time is nearly constant (22.68 s) and does not affect the consensus overhead of PMAB.

ConsensusTime. As shown in Figure 10, we test the time variation of PMAB's *AuditConsensus* per round as the

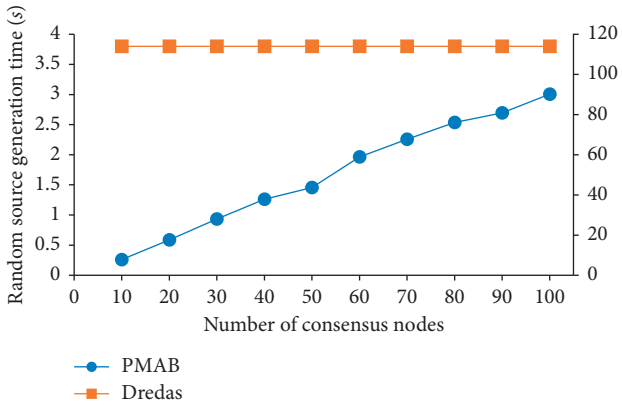


Figure 8: The random source generation time comparison in *RandomGenTime*.

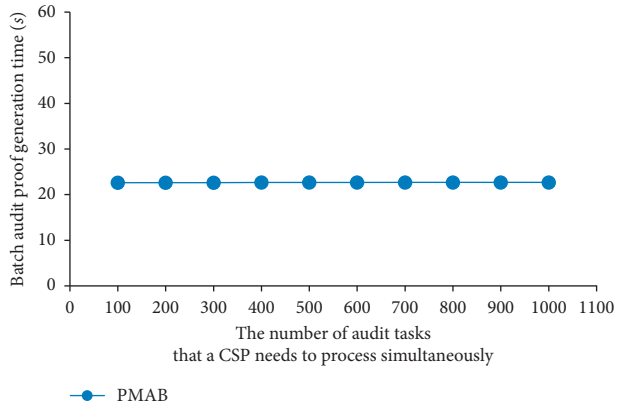


Figure 9: The computation cost in *ParallGenProof*.

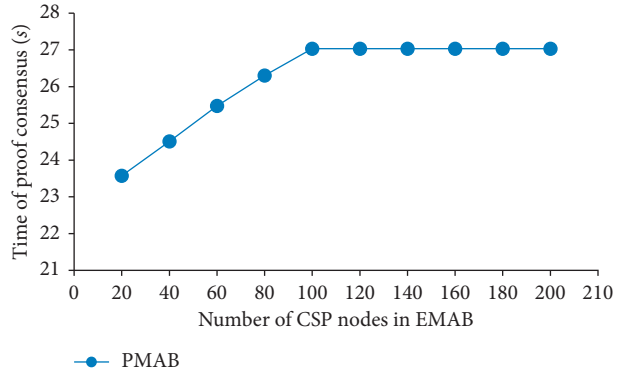


Figure 10: The communication cost in *ConsensusTime*.

number of CSPs increased (from 20 to 200). Each CSP node has 1000 audit tasks and each audit task challenged 250 data blocks. It is obvious that the time of PMAB's audit consensus tends to be constant (27.03 s) with the increase of the number of CSPs, since only a limited number of consensus nodes are needed to complete the audit consensus (the number of consensus nodes in this experiment is limited to less than 100). Therefore, PMAB has strong scalability and stability.

7. Conclusions

In this paper, PMAB for outsourced data in cloud storage was proposed. In PMAB, in order to achieve the goal of automatic audit safely, we constructed an audit chain architecture based on Ouroboros [20] and an incentive mechanism based on credit to allow CSPs to audit each other mutually with anticollusion. In addition, an audit protocol was designed to achieve public audit efficiently with low cost of audit verification. Security and performance analyses showed that PMAB achieves great audit efficiency and security goals. In future work, we will aim to research more specific incentive mechanism quantitative design and efficient problem positioning in batch audit.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (no. 2018YFB0803900), the National Natural Science Foundation of China (nos. 92067103 and 61772403), the Key Research and Development Program of Shaanxi (no. 2021ZDLGY06-02), the Key Scientific Research Program of Education Department of Shaanxi (no. 20JY015), the Fundamental Research Funds for the Central Universities (no. JBF211502), the Natural Science Foundation of Shaanxi Province (no. 2019ZDLGY12-02), the Natural Science Basic Research Plan in Shaanxi Province of China (no. 2020JM-184), 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.

References

[1] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.

[2] X. Jia, D. He, Q. Liu, and K.-K. R. Choo, "An efficient provably-secure certificateless signature scheme for internet-of-things deployment," *Ad Hoc Networks*, vol. 71, pp. 78–87, 2018.

[3] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: a survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[4] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6G networks: new areas and new challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.

[5] O. Gireesha, N. Somu, and K. Krithivasan, "IIVIFS-WASPAS: an integrated Multi-Criteria Decision-Making perspective for

cloud service provider selection," *Future Generation Computer Systems*, vol. 103, pp. 91–110, 2020.

[6] A. 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*, pp. 1–10, ACM, Istanbul, Turkey, September 2008.

[7] C. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transaction on Information and System Security*, vol. 17, no. 4, pp. 15.1–15.29, 2015.

[8] 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.

[9] Y. Zhu, G. J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

[10] H. Tian, Y. Chen, C.-C. Chang et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2015.

[11] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 99, pp. 2402–2415, 2017.

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

[13] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for iot data," in *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*, pp. 468–475, IEEE, Honolulu, HI, USA, June 2017.

[14] 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.

[15] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

[16] P. Huang, K. Fan, H. Yang, K. Zhang, and Y. Yang, "A collaborative auditing blockchain for trustworthy data integrity in cloud storage system," *IEEE Access*, vol. 99, p. 1, 2020.

[17] N. Lu, Y. Zhang, W. Shi, S. Kumari, and K. K. R. Choo, "A secure and scalable data integrity auditing scheme based on hyperledger fabric," *Computers & Security*, vol. 92, Article ID 101741, 2020.

[18] K. Fan, Z. Bao, M. Liu, A. V. Vasilakos, and W. Shi, "Dredas: decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial Iot," *Future Generation Computer Systems*, vol. 110, pp. 665–674, 2019.

[19] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proceedings of the 2019 ACM Conference on Economics and Computation*, pp. 459–473, Phoenix, AZ, USA, June 2019.

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357-358, Springer, Berlin, Germany, 2017.

[21] M. Stadler, "Publicly verifiable secret sharing," *Advances in Cryptology—Eurocrypt'96*, pp. 190–199, Berlin, Germany, 1996.

[22] J. C. N. S. Kenneth and G. Paterson, "Efficient identity-based signatures secure in the standard model," in *Information Security and Privacy*, pp. 207–222, Springer, Berlin, Germany, 2006.

[23] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on Ethereum systems security," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–43, 2020.

[24] J. Dean and S. Ghemawat, "MapReduce," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.