Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.030191
Article

Tech Science Press

Blockchain-Based Privacy-Preserving Public Auditing for Group Shared Data

Yining Qi[1,2,*], Yubo Luo[3], Yongfeng Huang[1,2] and Xing Li[1,2]

[1]Tsinghua University, Beijing, 100084, China
[2]Beijing National Research Center for Information Science and Technology, Beijing, 100084, China
[3]University of North Carolina at Chapel Hill, North Carolina, 27599, USA
*Corresponding Author: Yining Qi. Email: qyn18@mails.tsinghua.edu.cn
Received: 20 March 2022; Accepted: 22 June 2022

Abstract: Cloud storage has been widely used to team work or cooperation development. Data owners set up groups, generating and uploading their data to cloud storage, while other users in the groups download and make use of it, which is called group data sharing. As all kinds of cloud service, data group sharing also suffers from hardware/software failures and human errors. Provable Data Possession (PDP) schemes are proposed to check the integrity of data stored in cloud without downloading. However, there are still some unmet needs lying in auditing group shared data. Researchers propose four issues necessary for a secure group shared data auditing: public verification, identity privacy, collusion attack resistance and traceability. However, none of the published work has succeeded in achieving all of these properties so far. In this paper, we propose a novel blockchain-based ring signature PDP scheme for group shared data, with an instance deployed on a cloud server. We design a linkable ring signature method called Linkable Homomorphic Authenticable Ring Signature (LHARS) to implement public anonymous auditing for group data. We also build smart contracts to resist collusion attack in group auditing. The security analysis and performance evaluation prove that our scheme is both secure and efficient.

Keywords: Provable data possession; data integrity; blockchain; ring signature

1 Introduction

Number of companies and developers turning to cloud service has been growing rapidly in recent years [1,2]. Other than individual users, these enterprise and corporate customers usually work in groups, sharing data with each other. What is more, this exchange of data may sometimes occur among entities across various domains for co-operated research, making use of cloud storage service.

One of the most compelling topics for cloud data is its integrity. Group sharing makes it even more complicate. One group member, which we call it data owner, generates some dataset and uploads it to cloud. The others can access the data and make use of it, which are called data users. To data users, they cannot ensure whether data corruption is due to hardware/software failure, human errors of data owner or some malicious attacks. What makes things even worse, cloud service providers may conceal corruptions

from users [3], to avoid their responsibilities and maintain business reputation. Thus, group data sharing brings in more variables for trust between members. A method for group data verification is much needed.

There has been a few of works focusing on checking data correctness in cloud storage. An intuitive approach easy to call up is to retrieve the datasets from cloud and verify some kind of signatures (e.g., RSA) or collision-resistant hash values (MD5, SHA-256, etc.). Even though this method and some improved works [4,5] fulfill the basic demands of data verification, it seems more and more unaffordable since the scale of cloud data has become too large in past few years [6,7]. So, a new idea has been proposed: data owners compute specially designed digital signatures (called "tags") for their data partitioned into blocks, and verifiers check the integrity of cloud data using tags and some intermediate results based on data blocks, which are offered by cloud storage. This method is called Provable Data Possession (PDP) [8].

Quite a few following works [9–12] paid attention to the topic of employing independent verifier (neither data owner nor cloud service provider) to execute the process of integrity proof verification, which is called public verification, or public auditing. Choosing public auditing may have various reasons, but there are two main ones: on one hand, most users turn to cloud service to save time and money, which means that they may have no enough computation and storage source in local devices, even no enough cryptographic knowledge necessary for executing PDP verification; on the other hand, private verification is a kind of unilateral verification. If user comes up with a negative result "the data is corrupted", cloud service may dispute it. Public verification can solve these two problems. Wang et al. [9] proposed a privacy-preserving solution which ensured that public verifiers cannot infer the original content of challenged data blocks. In other words, it enables zero-knowledge integrity proving. This point is especially important when public verifier is Third Party Auditor (TPA), or any other data user to whom the owner has not decided to disclose the data. Based on this work, some researchers [13–18] continued to improve various aspects including application scenarios and security. Unfortunately, few works notice the significant difference between personal data and group shared data.

The most striking feature of group data is its sharing among multiple users. Although the basic public auditing scheme can be extended to group scenario, there exists an often-ignored privacy issue involved in data sharing, which we call it identity privacy. To illustrate what is identity privacy, we take an instance as follows. Alice and Bob work together in the same group, relying on cloud service to share data. The shared data is organized following some certain pattern, such as blocks or records. Alice generates a data block m_i and becomes its owner. She signs the block with her private key and generates a tag necessary for data auditing. The data user, Bob, wants to make sure whether the data block is intact before downloading it. He challenges the block and authorizes a public verifier to check the integrity proof from cloud. Then, according to the basic auditing scheme, the verifier should use the public key of data owner for proof verification. What is more, if Bob modifies a data block, he should resign the block and upload the new tag. Next time, public verifiers will use his public key to check the integrity of this block. It can be easily inferred that a public verifier is able to obtain the information about the ownership correlation between data blocks and group members. Furthermore, as time goes on, a public verifier will eventually learn the details of which data block is accessed and modified by a certain user, since public keys reveal identities of group members. Finally, the ownership and behavior of group members, a necessary part of privacy, are undoubtedly leaked to public verifiers.

This privacy leaking may bring in unpredictable consequences. Considering a scenario of data assets transaction, Alice and Bob are two entities exchanging their data. Bob wants to know whether the data he will buy is intact. Due to the specificity of data assets, Alice needs a convincing integrity verification scheme that does not require Bob to download the challenged data blocks. It seems that the best way is to

turn to a public verifier. However, those auditing schemes with no identity privacy preserving, will reveal valuable information of this transaction: who at which time tries to access whose data.

Wang et al. focused on group auditing and proposed a series of schemes [19,20] to solve the problem. They firstly proposed Knox [19], a group data auditing scheme based on group signatures. Knox relies on a "group manager" to control key generation and signature tracking for all users. It is no doubt that this centralized administrator will raise concerns about key and privacy disclosure. To remedy this defect, again the same authors proposed Oruta [20], using ring signature in the process of generating tags, in order to hide the identity of data owner from public verifier. Thus, we can say this scheme has enabled anonymous data auditing.

However, Oruta ignored an important issue in group data sharing: how to prevent members from malicious activities. Anonymous auditing brings another serious problem, that group members may deny some harmful or incorrect data they uploaded. What is more, when some member quit group, there needs solution to dispose of its data blocks. Since the identities of data owners are kept in secret, we need a convincing mechanism to make sure the quit member is indeed the data owner, not just on one side of the story.

Recently, blockchain-based PDP schemes draw attention of researchers. Han et al. [21] noticed the risk brought by untrusted TPA, but they made a mistake in the smart contract design that allowed CSP to perform replay attacks. Yuan et al. [22] introduced identity-based cryptography into blockchain-based PDP scheme, to avoid the complex certificate management caused by the public key infrastructure. Li et al. [23] tried to add cryptocurrencies to PDP scheme to promote TPA to be more active. And work [24] made some fine attempts in cloud-edge computation scenario. These works proved that blockchain-based PDP is practical, but lacked a full-featured solution to solve the problem of group data auditing.

To solve aforementioned problems, we design a novel blockchain-based anonymous public auditing scheme for group shared data. We choose ring signature for block tag construction to achieve fully privacy-preserving data auditing, and smart contracts to reduce manual intervention and possible repudiation. We utilize distributed ledger to store block tags for better reliability, and more impartial verification.

Contribution. 1) We introduce a construction of blockchain-based PDP scheme, which enables anonymous integrity verification via smart contract. 2) We propose a novel linkable ring signature tag generation method to protect identity privacy. 3) We analyze the security of our proposed scheme and evaluate its performance.

2 Problem Statement

In this section, we firstly describe the framework of cloud-chain integrated auditing system and its security model. Then aiming at the requirements for low performance devices network and security threat, we propose design goals to instruct the following scheme construction. Finally, we give the description of some preliminaries.

2.1 System Model

A system model of blockchain-based group data auditing is shown in Fig. 1. There are three kinds of entities and a blockchain network involved in group data auditing: a group of users, cloud service provider and public verifiers. Different from the work of [14], our scheme does not need to divide users into the original one and newcomers. All the users work in one group and each of them is allowed to access the shared data. Public verifier, role often taken by third-party auditor, can also be any other entities that have no interests involved in data to be audited. Cloud service provider offers storage for

shared data, but not like previous works, it does not store meta data used for verification. Distributed ledger of blockchain network take over this duty instead. All of the three entities join in blockchain network and manage shared data via smart contracts.

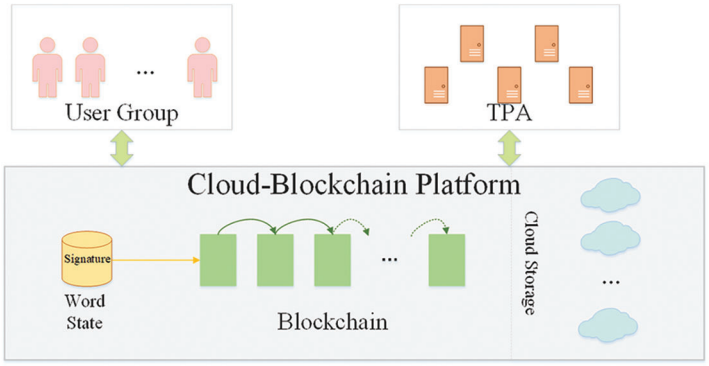

Figure 1: Tag generation time

The process of data auditing scheme in blockchain network is as follows. A group member send an auditing request to cloud service provider by invoking functions defined in smart contracts. The public verifier, playing the role of endorsement node in blockchain network, validate the request, including whether challenged data exists. After auditing request is accepted by blockchain network, cloud service provider compute integrity proof and send it by invoking smart contracts. The submission of integrity proof is wrapped as some kind of blockchain transaction, thus its verification can be naturally imbedded in the process of endorsement. Once public verifiers verify the integrity, they attach their endorsement to the proof, promoting the whole network accept it. In this way, the challenge-response PDP scheme has been transformed into the execution process of smart contracts.

2.2 Threat Model

Except the most basic threats of data corruption as in single user scenario, we also notice that there are other risks particularly lying in the situation of group data auditing.

Collusion Attack. Multiple-user structure of group members bring in new risks rooted in interactions among different entities. Entities participating in data auditing may collude to achieve certain purpose. For instance, data user may collude with verifiers to repudiate cloud storage or data owner, slandering the status of data stored in cloud. On the contrary, cloud service provider and public verifiers can also be complicit in concealing data corruption. Furthermore, data owner may speak on the side of cloud service provider, trying to disprove the adverse verification results made by verifiers and get users to accept its data.

Privacy Leakage. First of all, the owner-member relationship of cloud data is a kind of privacy to the group. Secondly, the identity of user who sends integrity auditing request, including meta data updating caused by data dynamics, should also be considered as secret and confidential information. Two aspects are both indispensable. Without proper identity shielding, a public verifier who engage in auditing work for the same group for a long time, can easily collect enough information to form activity records of every group member. These records can reveal which member accesses and modifies what kind of data at which time, and is able to allow a malicious attacker to distinguish valuable targets.

2.3 Design Goals

Taking aforementioned key points into account, we plan to design a blockchain-based PDP scheme with following features:

1) Public Auditing: A public verifier is able check data integrity without retrieving any or part of the original content from cloud storage.
2) Non-repudiation: The final decision on integrity verification is impartial and authentic. That is to say, the possibility of successful arbitrary manipulation of auditing results via collusion attack is negligible.
3) Identity Unforgeability: Valid meta data can only be generated by group member, using correct private and public keys. Any external or internal parties trying impersonate some group member cannot pass the integrity verification.
4) Privacy-Preserving: Public verifier cannot reveal owner identity of the data blocks it checks, or the identity of group member who requires the auditing.

3 A New Ring Signature Scheme

On the basis of previous works [9], we found that tags, the most important component of meta data for data integrity verification, are essentially various digital signatures that all have the property of zero-knowledge in common. The reason that most previous schemes failed to support privacy-preserving group auditing is the signature schemes adopted have no corresponding mechanism. Therefore, we choose ring signatures to protect ownership privacy and sensitive activity information for user group.

Unfortunately, most ring signature schemes are not suited for directly used in public verification. Because these schemes are usually designed for message authentication, whose key difference from those for public verifiable PDP is the absence of support for blockless verifiability. Directly sending data blocks to auditors is not only inefficient for communication overhead, but also contrary to the purpose of privacy preserving.

Taking the aforementioned factors into consideration, we design a new linkable homomorphic authenticable ring signature (LHARS) scheme adapted to group shared data PDP. It is extended from a classic ring signature scheme [25]. In the following chapters, we will introduce how the new scheme supports properties necessary for privacy-preserving public PDP via some critical alteration.

3.1 Preliminary

Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups of prime order q and $g_1,\ g_2$ be their generators respectively. Bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ holds properties as follows:

Bilinearity: for all $u \in \mathbb{G}_1,\ v \in \mathbb{G}_2,\ a,\ b \in \mathbb{Z}_q$, there holds $e(u^a,\ v^b) = e(u,\ v)^{ab}$.

Non-degeneracy: $e(g_1,\ g_2) \neq 1$.

Computability: there exists an efficient algorithm for computing mapping e in polynomial time.

Security Assumptions

Our scheme is built on two important security assumptions as follows:

Computational Diffie-Hellman Assumption: Consider a cyclic group $\mathbb{G}$ of prime order q. Choose g as a random generator of $\mathbb{G}$ and random elements $a,\ b$ from $\mathbb{Z}_q$. Given $(g,\ g^a,\ g^b)$, it is computationally intractable to compute value of g^{ab}.

Discrete Logarithm Assumption: Consider a cyclic group $\mathbb{G}$ of order q. For any random element $a,\ b$ of $\mathbb{G}$, an integer $k \in \mathbb{Z}_q$ that solves the equation $b^k = a$ is termed a discrete logarithm. If q is a sufficiently large prime number, computing k in polynomial time is hard.

Linkable Ring Signature

Signature generated in blockchain network has been proved to be practical by work [26,27]. But not all kinds of digital signatures are proper to design a PDP scheme. Ring signature was firstly introduced in 2001, whose name comes from its ring-like structure of signature algorithm. Ring signature is a kind of digital signature that can be performed by any member in a set of users connected by shared keys. And verifiers can learn whether the signature comes from certain set of users by checking its validity. One of the most distinctive securities of ring signature is that it is highly computationally infeasible to determine the real signer identity. The member group used to sign ring signature can be easily set without additional setup, which makes the scheme well suited for ad hoc group.

High anonymity brings in another problem: for a user, how to prove a certain signature is signed by itself without breaking the anonymity? Linkable ring signature solves the problem. Linkability means, two signatures signed by the same user can be proved to have some kind of link, which is called the signatures are linked.

3.2 Constructions of LHARS

The construction of LHAR consists of three components: KeyGen, SigSign, SigVerify.

KeyGen is the setup phase of whole scheme and each member of group should invoke it to generate their own private and public keys. SigSign is the algorithm used by user to generate signature for data block, with user private/public keys and a ring extracted from all the public keys of group members. SigVerify helps a public verifier not only to check the integrity of a data block, but also whether its signature is signed by a group member. Algorithms of LHARS build the basis of privacy-preserving PDP for group shared data, which will be introduced in the next section.

KeyGen. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups of prime order q and $g_1,\ g_2$ be their generators respectively. Let e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be bilinear map. Choose two different collision-resistant hash functions that fulfill H_1: $(0,\ 1)^* \rightarrow \mathbb{Z}_q$ and H_2: $(0,\ 1)^* \rightarrow \mathbb{G}_1$. The group chooses a private-public key pair for submission as $\pi \leftarrow \mathbb{Z}_q,\ \varrho = g_2^{\pi}$.

For each user u_i, pick random element $x_i \leftarrow \mathbb{Z}_q$ as its own private key, and compute the public key following $y_i = g_1^{x_i}$.

SigSign. Denote the total number of current members in group as d. The public keys of group members are $(y_1,\ y_2,\ \ldots,\ y_d)$. For a data block m, owner u_j computes ring signature in the following way:

1) Pick $(n-1)$ public keys randomly from all the group members to form a n-member key ring $L = (y_1,\ y_2,\ \ldots,\ y_n),\ 1 \le j \le n$.
2) Compute $h = H_2(L)$. And $\tilde{y} = h^{x_j}$.
3) Choose random element $\lambda \leftarrow \mathbb{Z}_q$, and compute

$$c_{j+1} = H_1(L,\ \tilde{y},\ g_1^{\lambda},\ h^{\lambda})$$

4) For other $u_i,\ i \neq j$ in ring L, pick random element $s_i \leftarrow \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ g_1^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i})$$

5) Finally, compute

$$s_j = \lambda - x_i c_i$$

$$t = (c_1 g_1^m)^{\pi}$$

The whole signature for data block m is $\sigma = (c_1,\ s_1,\ \ldots,\ s_n,\ \tilde{y},\ t)$.

SigVerify. For a given data block m, corresponding ring L and signature $\sigma = (c_1,\ s_1,\ \ldots,\ s_n,\ \tilde{y},\ t)$, a public verifier checks the validity of signature as follows:

Reconstructs part of the signatures: when $1 \le i \le n-1$, compute

$$z_i' = g_1^{s_i} y_i^{c_i},\ z_i'' = h^{s_i} \tilde{y}^{c_i},\ c_{i+1}' = H_1(L,\ \tilde{y},\ z_i',\ z_i^{''})$$

After that, compute $\tilde{c}_1 = H_1(L,\ \tilde{y},\ z_n',\ z_n'')$.

Then check the equation $e(\tilde{c}_1 g_1^m,\ \varrho) = e(t,\ g_2)$. If it holds, accept the signature; otherwise, reject.

3.3 Security of LHARS

Denition 1 (Unforgeability). Let $\mathcal{RSO}(L,\ m)$ be a signing oracle that accept any inputs as public keys $L = (y_1,\ y_2,\ \ldots,\ y_n)$ and block m, produces ring signature σ which is checked by the signature verification algorithm SigVerify. Denote the verification as $\mathcal{V}(L,\ m,\ \sigma)$. An LHARS signature scheme is unforgeable if, for any PPT (Probabilistic Polynomial-Time) adversary $\mathcal{A}$ with a chosen ring L, construct inequation

$$Pr\big(\mathcal{V}(L,\ m,\ \sigma) = 1;\ \mathcal{A}^{\mathcal{RSO}}(L)\big) \le \epsilon(k)$$

For any polynomial ϵ, the inequation holds. That is to say, with sufficient large prime order, the probability of forging a LHARS signature is negligible.

Definition 2 (Signer Ambiguity). An LHARS signature scheme is signer ambiguous if for any PPT algorithm $\mathcal{E}$, on inputs of any message m, any list L of n public keys, any set of t private keys $\mathcal{S} = (\hat{x}_1,\ \hat{x}_2,\ \ldots,\ \hat{x}_t)$, as well as any valid signature σ on L and m signed by user u_j, the probability of extracting user u_j from σ is

$$Pr[\mathcal{E}(L,\ m,\ \mathcal{S},\ \sigma) \to u_j] = \begin{cases} \epsilon\left(\dfrac{1}{n-t} - \dfrac{1}{Q(k)},\ \dfrac{1}{n-t} + \dfrac{1}{Q(k)}\right),\ \hat{x}_j \notin S \text{ and } 0 \le t < n-1 \\ > 1 - \dfrac{1}{Q(k)},\ o.w. \end{cases}$$

for any polynomial $Q(k)$.

The formula above implies an important fact: if the private key of real signer has not been leaked, its identity is as safe as any other undisclosed member; key exposures of other members will not directly reveal the signer identity, but weaken its security.

Definition 3 (Linkability). Let $L = (y_1,\ y_2,\ \ldots,\ y_n)$ be a list of n public keys, $(m_1,\ \sigma_1),\ (m_2,\ \sigma_2)$ are two messages with valid signatures respectively generated on L. Denote the signers of $\sigma_1,\ \sigma_2$ are $u_1,\ u_2$. An LHARS scheme is linkable if there exist a PPT algorithm $\mathcal{F}$, that outputs 1/0 with probability

$$Pr[\mathcal{F}(m_1,\ m_2,\ \sigma_1,\ \sigma_2) = 0 : u_1 = u_2] \le \epsilon(k)$$

$$Pr[\mathcal{F}(m_1,\ m_2,\ \sigma_1,\ \sigma_2) = 1 : u_1 \ne u_2] \le \epsilon(k)$$

for all sufficiently large k, $u_1,\ u_2 \in L$.

4 Blockchain-Based Privacy-Preserving Public Auditing Scheme

4.1 Reliable Signature Storage

An issue that has not been addressed in the past is that, since signature on block is the main and unique evidence for integrity verification, how to ensure retrieving authentic block signature concerns the security of

the whole public PDP scheme. In traditional PDP schemes [9], signatures are often stored in cloud storage. When checking integrity, the cloud service provider extracts signatures from its own storage. Unfortunately, just like any centralized solution, cloud signature storage also suffers from data corruption. What is more, encountered with verification failure result, cloud service provider may argue that this is only the consequence of corrupted signature.

To ensure the correctness of signatures and support data dynamics, previous works adopt Merkle Hash Tree (MHT) to manage the corresponding relationship between blocks and signatures. Maintaining the additional data structure brings in complex imbalance problem. Similarly, using MHT stored in cloud cannot determine whether it is the data block or the signature corrupted.

To solve this problem, we exploit a novel blockchain-based solution, which makes use of distributed ledger. Blockchain platforms such as Hyperledger Fabric offer two kinds of storage: one is the sequenced, tamper-resistant record of all the transactions in the past; the other is so-called world-state database that maintains the current state of digital asset. In our proposal, we define a signature as the "state" of responding block. The world-state storage keeps the signatures of every block in the form of key-value database. In our proposal, data operations are wrapped as transactions which lead to transitions of signatures. The sequenced records of blockchain transactions have immutability so that the states of blocks, as the results of such transitions, is also authentic. Tracing back a block to its credible operation records can help reconstruct its signature. In this way, we ensure that blockchain-based signature storage is reliable.

Discussion. Revisit the structure of an LHARS signature $\sigma = (c_1, s_1, \ldots, s_n, \tilde{y}, t)$ on block m. Consider that m is an element of $\mathbb{Z}_q$ and σ contains $n+2$ elements of $\mathbb{G}_1$, one element of $\mathbb{G}_2$ where $\mathbb{G}_1$, $\mathbb{G}_2$ are cyclic groups with order q. That is to say, the size of every LHARS ring signature is $(n+3) \times |q|$ bits. But a closer observation will reveal that the part $(c_1, s_1, \ldots, s_n, \tilde{y})$ of signature only relies on the public key ring and random element λ chosen by signer so that different blocks uploaded in the same assignment can share the common part. Denote the number of blocks sharing the same $(c_1, s_1, \ldots, s_n, \tilde{y})$ as k_p, and the size of ring signatures can be reduced from $k_p(n+3) \times |q|$ bits to $(n+2+k_p) \times |q|$ bits.

4.2 Construction of Our Proposal

In this part, we will present our proposed new privacy-preserving group PDP scheme in details, not only the algorithms, but also the process of executing smart contracts. Our scheme consists of five modules: KeyGen, SigGen, Update, ProofGen, ProofVerify.

The relationship of different modules in blockchain-based group PDP scheme is shown in Fig. 2. Each member of group should invoke KeyGen to generate their own private and public keys. Before uploading a data block, data owner uses SigGen for generating ring signature (called "tags"). The algorithm of Update is the complement of SigGen, for deleting or adapting data blocks. The tags, used as evidence for checking integrity proof, are stored in blockchain ledger. ProofGen is algorithm operated by cloud service provider in order to generate aggregated signature proof in response to challenge. Public verifiers can check the correctness of proof by invoking ProofVerify. There are also request and informing modules in the figure, which are system service used by user and CSP provided by blockchain platform. And we omit their details in the following construction.

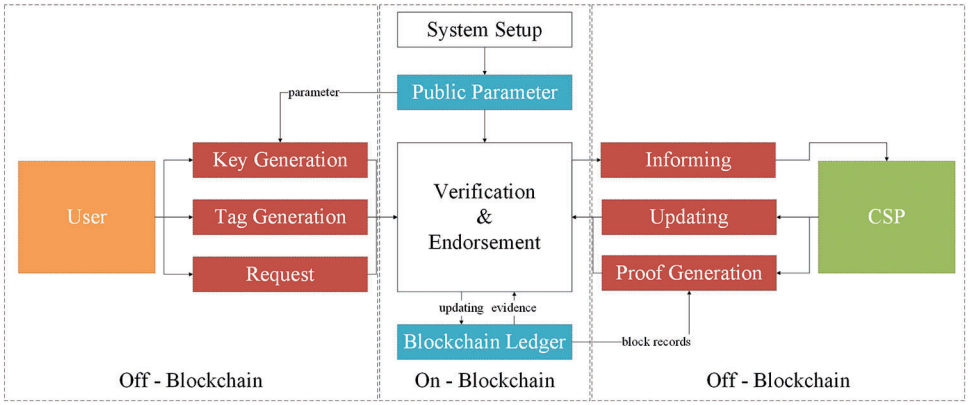

Figure 2: Modules of blockchain-based group PDP scheme

- *Basic Verification*

KeyGen. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups of prime order q and $g_1,\ g_2$ be their generators respectively. Let $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be bilinear map. Choose two different collision-resistant hash functions that fulfill $H_1: (0,\ 1)^* \rightarrow \mathbb{Z}_q$ and $H_2: (0,\ 1)^* \rightarrow \mathbb{G}_1$. The group chooses a private-public key pair for submission as $\pi \leftarrow \mathbb{Z}_q,\ \varrho = g_2^{\pi}$

For each user u_i, pick random element $x_i \leftarrow \mathbb{Z}_q$ as its own private key, and compute the public key following $y_i = g_1^{x_i}$.

SigGen. Denote the total number of current members in group as d. The public keys of group members are $(y_1,\ y_2,\ \ldots,\ y_d)$. Given a data block m to be uploaded, its owner u_j computes its ring signature in following way:

1) Pick $(n-1)$ users randomly from all the group members to form a n-member ring $L = (u_1,\ u_2,\ \ldots,\ u_n),\ 1 \le j \le n$.
2) Compute $h = H_2(L)$. And $\tilde{y} = h^{x_j}$.
3) Choose random element $\lambda \leftarrow \mathbb{Z}_q$, and compute

$$c_{j+1} = H_1(L,\ \tilde{y},\ g_1^{\lambda},\ h^{\lambda})$$

4) For other $u_i,\ i \neq j$ in ring L, pick random element $s_i \leftarrow \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ g_1^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i})$$

5) Finally, compute

$$s_j = \lambda - x_i c_i$$

$$t = (c_1 g_1^m)^{\pi}$$

Finally, the signature for data block m is $\sigma = (c_1,\ s_1,\ \ldots,\ s_n,\ \tilde{y},\ t)$.

ProofGen. Considering the case of challenging K blocks with indices $(idx_1,\ idx_2,\ \ldots,\ idx_K),\ K \ge 1$, generate aggregated zero-knowledge proof. The procedures follow the steps below:

1) Challenger picks K random element $(\gamma_{idx_1},\ \gamma_{idx_2},\ \ldots,\ \gamma_{idx_K})$ from $\mathbb{Z}_q$. Assemble the challenge $\{(idx_k,\ \gamma_k)\}_{1 \le k \le K}$.
2) Prover computes proof as follows:

According to the number of challenged blocks, choose random element $r \leftarrow \mathbb{Z}_q$ and compute $R = g_2^r$, in order to masking blocks.

Aggregate the blocks as

$$\mu = \gamma_{idx_k} \cdot m_{idx_k} + r$$

Then send $(\mu,\ R)$ as integrity proof.

ProofVerify. After receiving proof, public verifiers check the integrity in the following way:

For each data block m_{idx_k}, verifier refers to its record and extracts corresponding ring $L_k = \{y_{i,k}\}_{1 \le k \le K}$.

Then it reconstructs part of the signatures: when $1 \le i \le n-1$, compute

$$z'_{i,k} = g_1^{s_{i,k}} y_{i,k}^{c_{i,k}},\ z''_{i,k} = h^{s_{i,k}} \tilde{y}^{c_{i,k}},\ c'_{i+1,k} = H_1(L_k,\ \tilde{y},\ z'_{i,k},\ z''_{i,k})$$

After that, compute $\tilde{c}_{1,k} = H_1(L_k,\ \tilde{y},\ z'_{n,k},\ z''_{n,k})$.

Finally, check the equation

$$e\left(\prod_{k=1}^{K} \tilde{c}_{1,k}^{\gamma_{idx_k}} \cdot g_1^{\mu},\ \varrho\right) = e\left(R \cdot \prod_{k=1}^{K} t_{idx_k}^{\gamma_{idx_k}},\ g_2\right) \tag{1}$$

If it holds, accept the proof; otherwise, reject.

- *Enable Anonymous Dynamic Verification via Linkability*

When we discuss how to "update" a block of group shared data, we can easily divide the scenarios into three cases: adding, deleting and modifying. Obviously, adding a block is in essence the same as uploading a new block, so the method of SigGen can be easily reused.

In cases of modifying, process of dealing with updated new block can also apply SigGen. However, either modifying or deleting has another factor to be considered: how to make sure whether users sending requests are indeed data owners? The authentication of ownership is indispensable for data security, or else any malicious members can arbitrarily change blocks of others. Taking that in account, we use linkability of our ring signature to identify whether the requestor is the real owner of data block.

Theorem X. (Linkability) Let $L = (u_1,\ u_2,\ \ldots,\ u_n)$ be a ring constitutive of n group members. Data tags based on ring signature are linkable if there exists a PPT algorithm $\mathcal{F}_1$ which outputs 1/0 with

$$Pr[\mathcal{F}_1(L,\ m,\ m',\ \sigma,\ \sigma') = 0 : u = u'] \le \epsilon(k)$$

and

$$Pr[\mathcal{F}_1(L,\ m,\ m',\ \sigma,\ \sigma') = 1 : u \ne u'] \le \epsilon(k)$$

for all sufficiently large k, any $u,\ u' \in (u_1,\ u_2,\ \ldots,\ u_n)$, any data block $m,\ m'$ and corresponding tags $\sigma,\ \sigma'$. ϵ is some negligible function for sufficiently large k.

The algorithm $\mathcal{F}_1$ outputs 1 when it deems linkability existing in two tags, otherwise it outputs 0.

In our proposal, for two valid data tags $\sigma,\ \sigma'$ of block $m,\ m'$, it is easy to construct $\mathcal{F}_1$ as judgement of whether $\sigma = (c_1,\ s_1,\ \ldots,\ s_n,\ \tilde{y},\ t),\ \sigma' = (c'_1,\ s'_1,\ \ldots,\ s'_n,\ \tilde{y}',\ t')$ holds $\tilde{y} = \tilde{y}'$. Since $\sigma,\ \sigma'$ share the same ring L, $\tilde{y} = \tilde{y}' = [H_2(L)]^{x_j}$ generated by the same user u_j must holds.

Modifying. Procedure of modifying a block can be seen as uploading a new block m' to replace the original m. To prove ownership of original block m, data owner only needs to offer the new tag σ' for

checking whether $\tilde{y} = \tilde{y}'$ holds. In this way, we ensure data sovereignty of owners without adding too much extra computation and communication overhead.

Deleting. There is no new block to be uploaded in the case of deleting. Therefore, data owner needs to generate a temporary signature to prove its ownership. Data owner should choose a random message m' and generate σ' following the method in SigGen. If $\tilde{y} = \tilde{y}'$ holds and σ' is a valid tag, the deleting request can be identified as coming from the real data owner.

ProofGen. When a group member tries to verify the integrity of some data blocks, there are two situations: a) the user wants to check some certain blocks; b) the user wants to learn the overall status of whole data blocks. No matter which situation, K blocks with indices $(idx_1, idx_2, \ldots, idx_K)$, $K \geq 1$ will be picked out to be challenged. Thus, their procedures follow the same steps:

1) Pick K random element $(\gamma_{idx_1}, \gamma_{idx_2}, \ldots, \gamma_{idx_K})$ from $\mathbb{Z}_q$. Assemble a challenge request $\{(idx_k, \gamma_k)\}_{1 \leq k \leq K}$ and send it to blockchain network.
2) When cloud service provider receive challenge, it computes integrity proof as follows:

According to the number of challenged blocks, choose random element $r \leftarrow \mathbb{Z}_q$ and compute $R = g_2^r$, in order to masking blocks.

Aggregate the blocks as

$$\mu = \gamma_{idx_k} \cdot m_{idx_k} + r$$

Then send (μ, R) as integrity proof to blockchain network.

ProofVerify. After receiving proof, public verifiers check the integrity in the following way:

For each data block m_{idx_k}, verifier refers to its own distributed ledger and extracts corresponding ring $L_k = \{y_{i,k}\}_{1 \leq k \leq K}$.

Then it reconstructs ring signatures of blocks: when $1 \leq i \leq n-1$, compute

$$z'_{i,k} = g_1^{s_{i,k}} y_{i,k}^{c_{i,k}},\ z''_{i,k} = h^{s_{i,k}} \tilde{y}^{c_{i,k}},\ c'_{i+1,k} = H_1(L_k, \tilde{y}, z'_{i,k}, z''_{i,k})$$

After that, compute $\tilde{c}_{1,k} = H_1(L_k, \tilde{y}, z'_{n,k}, z''_{n,k})$.

Finally, check the Eq. (1). If it holds, accept the proof; otherwise, reject.

5 Security and Performance Analysis

In this section, we evaluate the performance of our proposed scheme. We perform a simulation instance of the proposed scheme based on Hyperledger Fabric. The instance is deployed on a Virtual Private Server (VPS) with CentOS 8_x64 system, using 1 CPU core@3.8 GHz, 2048MB RAM and 55GB SSD. The smart contracts are implemented by JAVA, with Pairing-Based Cryptography library (jPBC@2.0.0) and Hyperledger Fabric SDK for JAVA (v1.4.1).

5.1 Security Analysis

In this part, we will discuss the security properties of our blockchain-based PDP scheme, including correctness, non-repudiation, unforeability and privacy preserving.

Theorem 3. (Correctness) The proposed scheme has the property of correctness. That is to say, when most auditors in blockchain network are honest, an integrity proof from CSP cannot pass verification unless the cloud holds correct data.

Proof. Firstly, we should prove that Eq. (1) holds, which builds the basis of ProofVerify. According to the preliminary knowledge, the right-hand side of Equation X can be deduced as follows:

$$
\begin{aligned}
e\left(g_1^{\mu}\cdot\prod_{k=1}^{K}\tilde{c}_{1,k}^{\gamma_{idx_k}},\ \varrho\right) &= e\left(\prod_{k=1}^{K}\tilde{c}_{1,k}^{\gamma_{idx_k}}\cdot g_1^{\gamma_{idx_k}\cdot m_{idx_k}+r},\ g_2^{\pi}\right)\\
&= e\left(g_1^{r}\prod_{k=1}^{K}(\tilde{c}_{1,k}\cdot g_1^{m_{idx_k}})^{\gamma_{idx_k}},\ g_2^{\pi}\right)\\
&= e\left(R\cdot\prod_{k=1}^{K}(\tilde{c}_{1,k}\cdot g_1^{m_{idx_k}})^{\pi\cdot\gamma_{idx_k}},\ g_2\right)\\
&= e\left(R\cdot\prod_{k=1}^{K}t_{idx_k}^{\gamma_{idx_k}},\ g_2\right)
\end{aligned}
$$

The last equation holds if and only if $\tilde{c}_{1,k}=c_{1,k},\ k=1,\ \ldots,\ K$ also holds. Thus, if a proof passes the checking of Eq. (1), we can ensure that both ownership (contained in $c_{1,k}$) and content (contained in m_{idx_k}) are correct.

Theorem 4. (Non-repudiation) When most auditors in blockchain network are honest, a minority of dishonest auditors control the end result of integrity verification.

Proof. We have proved the correctness of checking integrity proof above. That is to say, if an auditor is honest, it can always draw a correct result. Since we define the process of checking proof as the validation of blockchain transactions, the authenticity of final decision relies on the mechanism of how blockchain network works. Blockchain network is believed to be tamper-free within less than 51% collusion because more honest auditors will give impartial judgements. Thus, we can say that the proposed scheme can stand up to attacks from a minority of dishonest auditors.

Theorem 5. (Unforgeability) For any member in group or cloud service provider, forging meta data of another member is infeasible in polynomial time if and only if the DL assumption holds.

Proof. Let $L=(y_1,\ y_2,\ \ldots,\ y_n)$ be a given ring of n group members. Assume a PPT adversary $\mathcal{A}$, able to make at most q_H times of queries to hash functions H_1 and H_2 as well as q_S times to $\mathcal{RSO}$, can forge ring signature σ with non-negligible probability as

$$Pr[\mathcal{A}(L)\rightarrow(m,\ \sigma):\mathcal{V}(L,\ m,\ \sigma)=1]>\frac{1}{Q(k)}$$

where Q is polynomial and k is the security parameter. $\mathcal{RSO}$ is a ring signature oracle which returns valid LHARS signatures upon queries of $\mathcal{A}$.

Now we assume that $\mathcal{A}$ constructs a PPT simulator $\mathcal{M}$ to generate forged signature. Since $\pi\leftarrow\mathbb{Z}_q,\ \varrho=g_2^{\pi}$ are the common keys shared by group members, we suppose that $\mathcal{M}$ holds $(\pi,\ \varrho)$ and simplify the problem as follows.

Ring Signing Oracle: Given any data block m, any public key list $L=(y_1,\ y_2,\ \ldots,\ y_n)$, the ring signing oracle $\mathcal{RSO}$ generate a signature. $\mathcal{M}$ simulates $\mathcal{RSO}$ to generate a signature without holding any secret keys of individual group members.

Without loss of generality, we assume that $\mathcal{M}$ randomly picks $r\leftarrow\mathbb{Z}_q$ and queries hash function to get $h=H_2(L)$. Then compute $\tilde{y}=h^r$ and chooses $c_1,\ \ldots,\ c_n,\ s_1,\ \ldots,\ s_n$. Back patch to

$$c_{i+1}=H_1(L,\ \tilde{y},\ g_1^{s_i}y_i^{c_i},\ h^{s_i}\tilde{y}^{c_i}),\ 1\le i\le n,\ n+1\rightarrow 1$$

Eventually $\mathcal{A}$ successfully forges $(c_1, \ldots, c_n, s_1, \ldots, s_n, \tilde{y})$ and $\mathcal{M}$ performs rewind-simulation to generate $(c_1^{'}, \ldots, c_n^{'}, s_1^{'}, \ldots, s_n^{'}, \tilde{y})$. Denote the forgery signer of $\mathcal{A}$ is j, then $\mathcal{M}$ can obtain x_j as follows:

$$c_{j+1} = H_1(L, \tilde{y}, g_1^{s_j} y_j^{c_j}, h^{s_j} \tilde{y}^{c_j})$$

$$c_{j+1}^{'} = H_1(L, \tilde{y}, g_1^{s_j^{'}} y_j^{c_j^{'}}, h^{s_j^{'}} \tilde{y}^{c_j^{'}})$$

Remember $\tilde{y} = h^r$, then

$$s_j + c_j x_j = s_j^{'} + c_j^{'} x_j$$

$$s_j + c_j r = s_j^{'} + c_j^{'} r$$

Solve and obtain

$$x_j = \frac{s_j - s_j^{'}}{c_j - c_j^{'}}$$

According to Literature [25], the probability of $\mathcal{M}$ to achieve a solution is at least $\left(\frac{1}{n(q_H + nq_S)Q(k)}\right)^2$, which is non-negligible. Therefore, once $\mathcal{A}$ is able to forge signature with an advantage of $\frac{1}{Q(k)}$, $\mathcal{M}$ is able to solve Co-CDH problem with an advantage of $\left(\frac{1}{n(q_H + nq_S)Q(k)}\right)^2$. Desired contradiction. Theorem is proved.

Theorem 6. (Privacy Preserving) If and only if DDHP (Decisinal Diffie-Hellman Problem) is hard, in the random oracle model, the probability of distinguishing signer of an LHARS signature is at most $\frac{1}{n}$, where n is the size of ring list L.

Proof. For any $g_1, h \in \mathbb{G}_1$, and $1 \leq j \leq n$, the distribution of $(c_1, \ldots, c_n, s_1, \ldots, s_n)$ is identical. Therefore, the probability of a PPT adversary $\mathcal{A}$ to distinguish c_j, s_j from $(c_1, \ldots, c_n, s_1, \ldots, s_n)$, in order to point out the signer u_j, is at most $\frac{1}{n}$. Literature [25] gives further detailed explanation.

5.2 Performance Analysis

To evaluate the performance of our scheme, we firstly compare the security properties of our proposal with other two comparable works Knox [19] and Oruta [20]. As shown in Tab. 1, our scheme supports various important properties including public auditing, identity privacy protection, collusion attack resistance and traceability. Though the previous works realized three of them, but no one has successfully integrated all of these before. Thus, we can say our scheme is the most comprehensive work ever done.

Table 1: Performance comparison between different group data PDP schemes

	Public auditing	Identity privacy protection	Collusion attack resistance	Traceability
Knox [19]		√		√
Oruta [20]	√	√		
Our proposal	√	√	√	√

To measure the practical performance of our scheme, we also deploy an instance on VPS, which has 1 CPU, 2GB memory and 2TB bandwidth. Based on platform provided by the open source blockchain project Hyperledger Fabric, we implement the chaincode via its SDK for JAVA. Fabric offers pluggable

consensus services and editable endorsement policies, as well as necessary membership services and certificate authority management. We configure the benchmark test tool Hyperledger Caliper to evaluate the performance of our blockchain-based PDP scheme. Caliper enables users to write the test and network configuration, then launch an instance and execute required smart contracts defined in given chaincode automatically.

We implement every on-chain algorithm in our scheme, wrapped as functions. All the functions are contained in different scripts written in Node.js, in order to complete various tasks in PDP auditing. When someone wants to execute a certain smart contract, it just needs to run the scripts.

Tag generation time is shown in Fig. 3. The red line with dot "●" is the time for computing one tag of our proposal, while the green line with "▪" is that of Oruta and the purple line with "▲" is for Knox. We control the size of group/ring which is necessary for generating group/ring signatures. The result shows that Oruta is no doubt the highest one among three works, and our proposal is less than a half of Knox when the size of ring is between 2∼10. Knox, as a group signature solution, has stationary performance, since its signature generation does not rely on the size of group. Our scheme, based on ring signature, is also growing linearly. It must be noted that, although the computation cost of ring-based solutions depends on the size of ring, it will not be limitless. Ring signature just need to select a ring with "proper" size to hide the identity of signer. The ring size in our comparison ranges from 2 to 10, which we think is enough to cover most cases in practice.

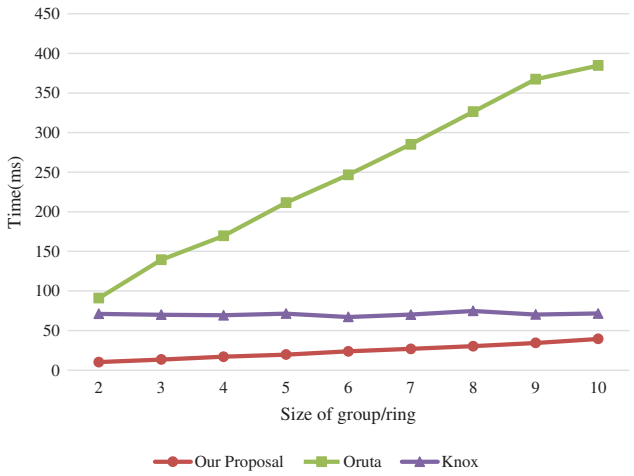

Figure 3: Computational cost for tag generation

Computational cost for integrity verification is shown in Fig. 4. The computing cost increases in member numbers. But the cost of per block is still very low. The red line with dot "●" is the time for computing one tag of our proposal, while the green line with "▪" is that of Oruta and the purple line with "▲" is for Knox. Since all the three schemes are based on bilinear pairing computation, there is no apparent difference in verification computation. There is a slight increase as the size of group/ring grows, but not very apparent. This is because bilinear pairings account for the bulk of the overhead.

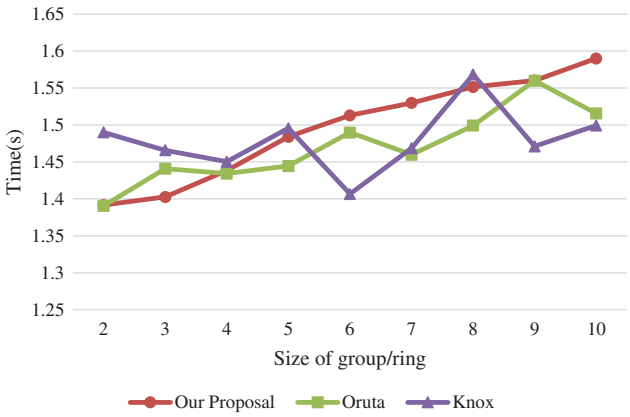

Figure 4: Computational cost for integrity verification

6 Conclusion

This paper focuses on exploring a blockchain-based PDP scheme for group shared data. We analyze the needs of group data auditing and find that resisting collusion attacks and identity privacy are the most important two issues. Blockchain network and smart contracts offer a potential solution of reliable outsourced computation, which makes tag generation and integrity verification free from collusion attack, and linkable ring signature provides support for anonymous data auditing. We design a novel method of LHARS scheme as well as a blockchain-based PDP scheme. Security analysis and performance evaluation prove that our scheme is both secure and efficient.

Funding Statement: The work is supported by the National Key Research and Development Program of China (No. 2018YFC1604002) and the National Natural Science Foundation of China (No. U1836204, No. U1936208, No. U1936216, No. 62002197).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Rudniy, "Data warehouse design for big data in academia," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 979–992, 2022.

[2] A. Berguiga and A. Harchay, "An IoT-based intrusion detection system approach for TCP syn attacks," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3839–3851, 2022.

[3] R. Jia, Y. Xin, B. Liu and Q. Qin, "Dynamic encryption and secure transmission of terminal data files," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1221–1232, 2022.

[4] M. Naor and G. N. Rothblum, "The complexity of online memory checking, " *Journal of the ACM*, vol. 56, no. 1, pp. 1–46, 2009.

[5] A. Oprea, M. K. Reiter and K. Yang, "Space-efficient block storage integrity," in *Proc. of 12th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, California, USA, 2005.

[6] J. Almutairi and M. Aldossary, "Exploring and modelling IoT offloading policies in edge cloud environments," *Computer Systems Science and Engineering*, vol. 41, no. 2, pp. 611–624, 2022.

[7] L. Jiang and Z. Fu, "Privacy-preserving genetic algorithm outsourcing in cloud computing," *Journal of Cyber Security*, vol. 2, no. 1, pp. 49–61, 2020.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner *et al.,* "Provable data possession at untrusted stores," in *Proc. of 14th ACM Conf.. Computer and Communication Security (CCS '07)*, Alexandria, Virginia, USA, pp. 598–609, 2007.

[9] 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, 2011.

[10] X. Tang, Y. Qi and Y. Huang, "Reputation audit in multi-cloud storage through integrity verification and data dynamics," in *Proc. of 2016 IEEE 9th Int. Conf. on Cloud Computing (CLOUD)*, San Francisco, California, USA, pp. 624–631, 2016.

[11] Z. Mo, Y. A. Zhou, S. G. Chen and C. Z. Xu, "Enabling non-repudiable data possession verification in cloud storage systems," in *Proc. of 2014 IEEE 7th Int. Conf. on Cloud Computing (CLOUD)*, Anchorage, Alaska, USA, pp. 232–239, 2014.

[12] H. Jin, H. Jiang and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 680–693, 2018.

[13] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang *et al.,* "MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.

[14] Z. Mo, Y. A. Zhou and S. Chen, "A dynamic proof of retrievability (POR) scheme with O(log n) complexity," in *Proc. of 2012 IEEE Int. Conf. on Communications (ICC)*, Ottawa, Canada, pp. 912–916, 2012.

[15] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu *et al.,* "Dynamic audit services for integrity verification of outsourced storages in clouds," in *Proc. of ACM Symp. on Applied Computing (SAC '11)*, TaiChung, Taiwan, pp. 1550–1557, 2011.

[16] Y. Zhu, H. Hu, G. J. Ahn and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[17] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717-1726, 2013.

[18] H. Tian, Y. Chen, C. C. Chang, H. Jiang, Y. Huang *et al.,* "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2017.

[19] B. Wang, B. Li and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. of Int. Conf. on Applied Cryptography and Network Security*, Berlin, Heidelberg, Springer, 2012.

[20] B. Wang, B. Li and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *In IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[21] B. Han, H. Li and C. Wei, "Blockchain-based distributed data dntegrity auditing scheme," in *Proc. of 2021 IEEE 6th Int. Conf. on Big Data Analytics (ICBDA)*, Xiamen, China, pp. 143–149, 2021.

[22] Y. Yuan, J. Zhang, W. Xu and Z. Li, "Identity-based public data integrity verification scheme in cloud storage system via blockchain," *the Journal of Supercomputing*, vol. 78, pp. 8509–8530, 2022.

[23] Y. Li, Y. Yu, R. Chen, X. Du and M. Guizani, "IntegrityChain: Provable data possession for decentralized storage," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1205–1217, 2020.

[24] Q. Mei, H. Xiong, Y. C. Chen and C. M. Chen, "Blockchain-enabled privacy-preserving authentication mechanism for transportation CPS with cloud-edge computing," *IEEE Transactions on Engineering Management*, in press, DOI 10.1109/TEM.2022.3159311.

[25] J. K. Liu, V. K. Wei and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Proc. of Australasian Conf. on Information Security and Privacy*, Sydney, Australia, pp. 325–335, 2004.

[26] C. Li, Y. Tian, X. Chen and J. Li, "An efficient anti-quantum lattice-based blind signature for blockchain-enabled systems," *Information Sciences*, vol. 546, pp. 253–264, 2021.

[27] C. Li, G. Xu, Y. Chen, H. Ahmad and J. Li, "A new anti-quantum proxy blind signature for blockchain-enabled internet of things," *Computer, Material & Continua*, vol. 61, no. 2, pp. 711–726, 2019.