Efficient Blockchain-Based Data Integrity Auditing for Multi-Copy in Decentralized Storage

Qingyang Zhang, Zhiming Zhang, Jie Cui, *Senior Member, IEEE*, Hong Zhong, Yang Li, Chengjie Gu, and Debiao He

***Abstract*—As the disruptor of cloud storage, decentralized storage could lead to a major shift in how organizations store data in the future. To ensure data availability, users generally encrypt the data and distribute it to multiple storage service providers. It is necessary to study data integrity verification in decentralized storage. Although some recent studies have proposed the using blockchain technology to assist auditing work in decentralized storage networks, the on-chain overhead still increases linearly with an increase in audit requests. Blockchain networks will inevitably be overloaded. In this study, we propose an efficient data integrity auditing scheme for multiple copies in decentralized storage. Particularly, using different polynomial commitment schemes, we first propose a basic scheme for verifying multiple copies of a single file, and then we propose an efficient batch auditing scheme for multiple copies of multiple files. Our scheme can significantly reduce the computation overhead of storage service providers while keeping the on-chain storage overhead constant. Security analysis and performance analysis show that our scheme is efficient and practical.**

***Index Terms*—Multi-copy, data auditing, efficiency, blockchain, decentralized storage, polynomial commitment.**

Manuscript received 14 December 2022; revised 29 June 2023; accepted 29 September 2023. Date of publication 9 October 2023; date of current version 24 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62272002 and 62202005, in part by the National Key R&D Program of China under Grant 2021YFB3100500, in part by the Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31, in part by the Natural Science Foundation of Anhui Province, China under Grants 2008085QF297 and 2208085QF198, and in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049. Recommended for acceptance by Q. Zheng. *(Corresponding author: Jie Cui.)*

Qingyang Zhang, Zhiming Zhang, Jie Cui, and Hong Zhong are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: qingyang.zhang.inchina@gmail.com; zzm627zzm@163.com; cuijie@mail.ustc.edu.cn; zhongh@ahu.edu.cn).

Yang Li is with the Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, Anhui 230037, China (e-mail: liyanghf@nudt.edu.cn).

Chengjie Gu is with the School of Public Security and Emergency Management, Anhui University of Science and Technology, Hefei 231131, China (e-mail: gcj@ustc.edu.cn).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shanghai Key Laboratory of Privacy Preserving Computation, MatrixElements Technologies, Shanghai 201204, China (e-mail: hedebiao@163.com).

Digital Object Identifier 10.1109/TPDS.2023.3323155

I. Introduction

SINCE the concept of cloud computing technology was proposed in 2006, several Internet companies, such as Amazon, Microsoft, and Alibaba, have commercialized cloud computing to provide services to users. As a basic cloud computing service, outsourced storage service attracts resource-constrained individuals and enterprises to outsource their local data to the Cloud Storage Provider (CSP), which can reduce users' expensive local hardware costs. Therefore, users can enjoy high-quality services at low costs. According to the IDC's forecast, by 2025, global data will grow to 175 ZB, and 49% of the world's stored data will reside in public cloud environments [1].

Simultaneously, it has brought numerous security challenges to storing data on the cloud. The CSP can delete a portion of infrequently used data to save storage costs, or data loss may occur when hardware fails. The data owner loses physical control of the data, making it difficult to verify the data integrity [2]. Therefore, it is crucial to verify the data integrity of the CSP. Most of the existing data integrity auditing schemes are based on Provable Data Possession (PDP) [3] and Proof of Retrievability (POR) [4] which have a similar idea. In simple terms, the user generates verification tags for the chunked data and stores them on the CSP. Then a Third-Party Auditor (TPA) initiates a random challenge to the CSP. After the CSP generates the corresponding certificate using the data and verification tags, the certificate is finally submitted to the TPA for verification. In particular, to prevent accidents and protect valuable data such as medical data, economic data, and scientific research data, it is necessary to generate multiple copies of the outsourced data. When one copy is lost or damaged, it can be recovered from the other copies. Cloud service companies usually establish a central cloud manager to manage each CSP. As a centralized manager, it usually distributes copies and integrates integrity proof during the auditing process. The existing multi-copy auditing schemes based on TPA mainly focus on efficiently verifying the integrity of all copies at once to reduce overhead.

However, these schemes also have some problems. 1) They are all based on the assumption that TPA is trusted. But, TPA is generally not trusted in reality and may even collude with CSP. [5] The above assumption is difficult to hold. 2) TPA will encounter the problem of a single point of failure. When the TPA fails, data integrity auditing will be unavailable. 3) These schemes assume that the cloud manager is honest and trustworthy and does not

1045-9219 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

collude with the CSP, which is unrealistic considering that the cloud manager and the CSP belong to the same organization [6].

With the innovation of blockchain technology, such as Bitcoin [7] and Ethereum [8], new ways of solving the above problems have emerged. Benefiting from the public, transparent and immutable characteristics of the blockchain, most existing blockchain-based schemes publish some auditing metadata on the blockchain and call it an audit log or audit trail. Furthermore, these schemes use smart contracts as auditors instead of untrusted TPA for integrity verification. Also, the development of blockchain technology brings novel storage models. Typically, Filecoin [9] is basically a blockchain built on top of the InterPlanetary File System (IPFS) [10] that aims to create a more potential decentralized storage market than cloud storage. Motivated by its incentive mechanism, the utilization of idle storage resources and communication resources is significantly improved. Naturally, in this more open storage model, an audit mechanism is needed to ensure data integrity. Some research on data integrity auditing in decentralized storage has gradually attracted attention.

However, there is a fatal problem with these existing methods. They use the blockchain as a transparent and immutable database to solve trust issues and enable public verification. When the scale of data becomes huge, the on-chain resources will be overload, particularly the storage overhead on the blockchain. Therefore, an efficient multi-copy auditing scheme is still required in the decentralized storage network.

A. Contributions

In summary, our contributions are as follows:

1) We propose a blockchain-based efficient data integrity auditing scheme for multi-copy in decentralized storage, utilizing homomorphic verifiable tag (HVT) and polynomial commitment to achieve an efficient performance.
2) We designed a general and efficient decentralized auditing paradigm that can significantly reduce the burden on the blockchain network. Our design consists of two schemes. The second scheme is based on the first scheme and effectively addresses its shortcomings. Specifically, our scheme can avoid linear growth of both computation overhead and on-chain storage overhead with the number of files when performing batch audit tasks. In particular, only a constant 128-byte on-chain storage overhead is required in each batch audit.
3) We demonstrate the security of the proposed scheme. Theoretical analysis and experiments show that the scheme is efficient and feasible in decentralized storage, and it can save blockchain resources significantly.

B. Outline

The rest of this article is organized as follows. In Section II, we outline the related work. Subsequently, we present preliminary and definitions in Section III. Next, we present a detailed description of the proposed scheme in Section IV. Security analysis and performance analysis are presented in Sections V and VI. Finally, we provide the conclusion of this work in Section VII.

II. Related Work

To make data more reliable and available, the storage of multiple copies is especially necessary. Lost or damaged copies can be recovered by relying on other intact copies. Curtmola et al. [11] proposed the first multiple replicas auditing scheme based on RSA signature. In their scheme, multiple copies of a file are generated by masking the encrypted file blocks with multiple random values so that different copies are distinguished and data privacy is protected. To support public verifiability, Barsoum et al. [12] presented two schemes for detecting the integrity of multiple copies, including the deterministic scheme and a probabilistic scheme. The latter is more practical and economical because a sensible method is to select some challenge blocks instead of all data blocks. To reduce the burden of certificate management, Li et al. [13] proposed a multi-copy auditing scheme by employing identity-based cryptography in multi-cloud storage.

In summary, most existing schemes are based on trusted TPA. This centralized third-party auditor will face some centralization problems such as single point of failure and performance bottlenecks. Besides, it is impossible to ignore the strong trust issue of TPA [14].

To solve the problems caused by centralization, the paradigm of decentralized storage based on blockchain is proposed. Verifying data integrity in decentralized storage networks has also become a new challenge and many auditing schemes have been proposed. Zhang et al. [15] proposed a blockchain-based auditing scheme. Untrusted TPAs need to publish audit logs on the blockchain. To solve the certificate management problem brought by public key infrastructure (PKI), Xue et al. [16] proposed an identity-based public auditing scheme. Their scheme also publishes audit logs on the blockchain, further unbiased, unpredictable, and unforgeable challenge set is generated by employing the nounce in the block header. Furthermore, a large amount of duplicate data in decentralized storage will lead to a waste of storage space. Xu et al. [5] proposed a blockchain-enabled deduplicatable data auditing mechanism. They build an index table in the blockchain that stores the hashes of the files. If the file uploader finds the same file hash value in the index table, the storage provider needs to receive the data ownership proof from the uploader and build a new file index table. Yuan et al. [17], Tian et al. [14] and Zhang et al. [18] also proposed blockchain-based secure deduplication auditing schemes which make use of convergent encryption to achieve block-level deduplication. Furthermore, Xu et al. [19] and Zhang et al. [20] implemented dispute arbitration and locate faults similarly utilizing smart contract. Du et al. [21] proposed an efficient auditing scheme by employing polynomial commitment. In the above three works, smart contract acts as a more trustworthy auditor to replace untrusted TPA in decentralized storage. In short, most existing schemes store the metadata during the audit process on the blockchain for public verification. As the number of audited

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

files increases, both the verification times and the total size of proof on the blockchain will grow linearly. And it would place a heavy burden on the blockchain network.

III. Preliminary and Definitions

A. Preliminary

1) Bilinear Maps: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of order q. Let g be the generator of $\mathbb{G}_1$. $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map with following properties:

- *Bilinear:* $\forall g, h \in \mathbb{G}_1$ and $a, b \in Z_p, e(g^a, h^b) = e(g, h)^{ab}$ holds.
- *Non-degeneracy:* $\forall g \in \mathbb{G}_1$, $e(g, g) \neq 1$, where 1 is the identity element.
- *Computability:* $\forall g, h \in \mathbb{G}_1$, there exists a effective algorithm to calculate $e(g, h)$.

2) Polynomial Commitment: Polynomial commitment scheme (PCS) is a cryptographic commitment scheme, first proposed by Kate et al. (KZG) [22]. Subsequently, Gabizon et al. proposed a batched version of the KZG scheme (PCS-PLONK) [23] that can verify an identical point for multiple polynomials. But the proof size and prover computation will grow linearly with the number of distinct points. To solve this problem, Boneh et al. proposed an efficient polynomial commitment scheme for multiple points and polynomials (PCS-MPAP) [24].

In brief, the PCS can be simplified as a two-party protocol with four phases: Setup, Commit, CreateWitness and VerifyEval. Among them, the CreateWitness phase and VerifyEval phase of these schemes are different. So we only give a brief description of these two phases.

- *Setup*$(\lambda) \rightarrow (pk, sk)$*:* Generate an appropriate algebraic structure and a public-secret key pair. It should be noted that the public key is $pk := (g, g^{\alpha}, \ldots, g^{\alpha^{s-1}})$, where g is a generator of group $\mathbb{G}$ and $\alpha \in Z_p$ is randomly chosen as the secret key.
- *Commit*$(pk, f(X)) \rightarrow C$*:* The prover makes a commitment for a polynomial $f(X)$ with coefficient vector $(c_0, c_2, \ldots, c_{s-1})$. By using pk, prover outputs the commitment $C = g^{f(\alpha)}$.
- *CreateWitness*$(pk, f(X), r) \rightarrow W$*:* Given the pk and the polynomial $f(X)$, the prover calculates the witness W on a random point r.
- *VerifyEval*$(pk, r, C, W) \rightarrow 0/1$*:* After receiving (C, W) from the prover, the verifier can check whether $f(r)$ is the evaluation of the polynomial $f(X)$ at point r.

3) Blockchain and Smart Contract: Blockchain technology is well known in the field of cryptocurrencies such as Bitcoin [7], Ethereum [8], and Filecoin [9]. Blockchain is essentially a shared, immutable, decentralized, and distributed ledger, which generally contains digital transactions, data records, and executables [25]. Many transactions are aggregated together to form blocks. After merging some block header data, the blocks are linked together by hashing to form a chronologically ordered chain of records. The consensus mechanism enables all nodes to agree on the content of the blockchain.

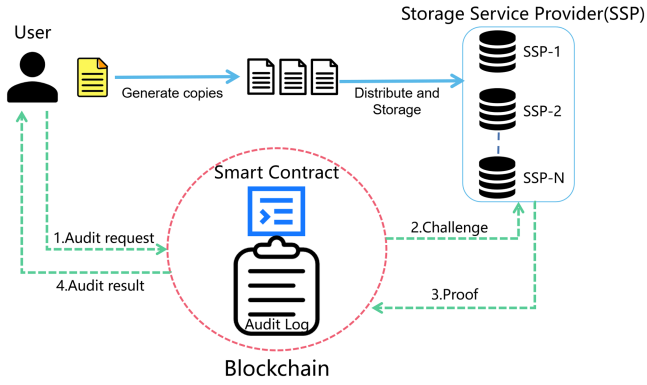


Fig. 1. System model of our scheme.

The Smart contract is a computer program or a transaction protocol that runs on the blockchain. It allows trusted transactions without third parties and is irreversible and immutable. To meet more application scenarios, Ethereum is the first platform to introduce the concept of smart contracts. Smart contract using Turing-complete programming languages brings more possibilities to the future of blockchain technology.

B. System Model

As depicted in Fig. 1, the system model of our proposed scheme has three entities: storage service provider, user, and blockchain. Their definitions are as follows:

- *Storage service provider (SSP):* SSP has a lot of storage resources and provides data outsourcing services for users. It also has the certain computing power to deal with data integrity auditing. SSP can be an individual or a cloud service provider.
- *User:* The user outsources local data to SSPs. He encrypts the original data into multiple different copies and stores the copies on different SSPs. After that, local data can be deleted to save storage space.
- *Blockchain:* The blockchain network is mainly composed of user nodes and miner nodes. Miner nodes can earn rewards by executing contracts or renting out storage resources. The smart contract is responsible for interacting with the SSPs and completing the integrity verification. The metadata of the audit process will be recorded on the blockchain to form an audit log.
- *System manager:* System manager is responsible for initializing some system parameters.

C. Security Model

Inspired by the POR security model [4], we define a security model for the proposed scheme, which contains four algorithms *(KeyGen, Setup, Prove, Verify)*.

- *KeyGen*$(\lambda) \rightarrow (pk, sk)$: Given the security parameter λ, the algorithm KeyGen generates the public key and private key pk, sk.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

- *Setup*$(F, sk, pk) \rightarrow (\hat{F}, \sigma)$: Given a file F and the keypair (pk, sk), the Setup algorithm produces the preprocessed file $\hat{F}$ and its corresponding tag σ.
- *Prove*$(\hat{F}, \sigma, pk, Chal) \rightarrow Proof$: Given the file $\hat{F}$, tag σ, public key pk and a random challenge $Chal$, the Prove algorithm outputs a proof $Proof$.
- *Verify*$(pk, Proof) \rightarrow 0/1$: Given the $Proof$ and pk, the Verify algorithm will output 0/1 representing reject and accept, respectively.

In this work, we would like the proposed scheme S = *(KeyGen, Setup, Prove, Verify)* to be correctness and soundness. Correctness requires that, for all (sk, pk), F, $\hat{F}$ and σ, the *Verify* algorithm accepts the valid $Proof$. For soundness, we define the security game of the proposed scheme as below.

- The challenger $\mathcal{C}$ runs the KeyGen algorithm to generate the keypair (pk, sk) and sends the pk to adversary $\mathcal{A}$.
- Given a file F chosen by $\mathcal{A}$, $\mathcal{C}$ runs the Setup algorithm to get the preprocessed file $\hat{F}$ and its corresponding tag σ. Then, $\mathcal{C}$ sends $(\hat{F}, \sigma)$ to $\mathcal{A}$.
- $\mathcal{C}$ chooses a random challenge $Chal$ that the challenge blocks have not been queried and sends $Chal$ to $\mathcal{A}$. Then, $\mathcal{A}$ generate a proof by an arbitrary algorithm $Art(\hat{F}, \sigma, Chal) \rightarrow Proof$.
- $\mathcal{A}$ wins the game if the $Proof$ generated by $\mathcal{A}$ can make $\mathcal{C}$ output 1 through the Verify algorithm.

Definition 3.1: The proposed scheme S is soundness if any probabilistic polynomial-time adversary $\mathcal{A}$ can win the above game with a non-negligible probability.

D. Algebraic Group Model

Fuchsbauer et al. [26] first proposed the algebraic group model (AGM) and proved the security of the BLS signature scheme [27] and Groth's zk-SNARK [28] in it. The AGM lies between the standard model and the Generic Group Model (GGM), and it is a restricted model of computation that covers group-specific attacks while allowing a meaningful security analysis. Besides, in their study, some variants of the standard Diffie-Hellman assumption (such as computational Diffie-Hellman and strong Diffie-Hellman) are actually equivalent to the Discrete Logarithm (DLOG) assumption in the AGM. Therefore the proof is simpler in the AGM than in the GGM. Subsequently, AGM was used to prove the security of PCS in zk-SNARK such as [23], [24], [29].

We call an an algorithm $\mathcal{A}_{alg}$ algebraic if whenever it outputs a group element $Z \in \mathbb{G}$, it also outputs a representation $\vec{z} = (z_1, \ldots, z_t)$ such that $Z = \prod_{i=1}^{t} L_i^{z_i}$ where $\vec{L} = (L_1, \ldots, L_t)$ is the list of all group elements given to $\mathcal{A}_{alg}$ during its execution so far.

By the security argument in [23], [24], we have the following theorem.

Theorem 1: PC-PLONK is knowledge soundness in the AGM assuming the q-discrete logarithm (Q-DLOG) assumption holds.

Theorem 2: PC-MPAP is knowledge soundness in the AGM assuming the Q-DLOG assumption holds.

TABLE I
NOTATIONS AND DESCRIPTIONS IN BASIC SCHEME

Notations	Descriptions
F	The original outsourced file
$Copy_i$	The ith copy of file F
E_K	AES symmetric encryption, K is the symmetric key
m_{ijk}	The sth sector of the jth block of the ith copy
σ_{ij}	The tag of the jth block of the ith copy
SSPi	The i-th SSP
$f_{m_{ij}}(X)$	A polynomial with s coefficients $\{m_{ijk}\}_{1 \leqslant k \leqslant s}$
$f_{M_i}(X)$	The challenge polynomial of the ith copy
$h_i(X)$	The quotient polynomial of the ith copy

IV. THE PROPOSED SCHEME

A. Overview

To clearly describe the proposed scheme, we assume a simplified storage scenario. An SSP stores only one copy and its tags and specific details will be shown in Section IV-B2. We emphasize that it can also be extended to real-world scenarios. We first propose a basic scheme. During the preprocessing and storage phase, the user generates different copies and corresponding tags for file F. However, PCS-PLONK used in the basic scheme will bring disadvantages to batch files audit tasks. During the auditing phase, the on-chain and off-chain overhead will grow linearly with the number of files. Therefore, we propose an efficient batch auditing scheme as an improvement. We use PCS-MPAP to improve the efficiency in the scenario of batch auditing multiple files. It should be noted that the second scheme can also be used for auditing scenarios of a single file. But when the number of files is 1, the basic scheme is more efficient. When the number of files is not less than 2, the efficiency of the second scheme has obvious advantages.

We would also like to emphasize that the two proposed schemes differ only in the auditing phase. They have the same setup phase and preprocessing and storage phase. In other words, SSP only needs to honestly store copies and corresponding tags to pass the integrity auditing of both schemes. To avoid repetition, we only describe the auditing phase in our second scheme.

B. Basic Scheme

1) Setup Phase: During the Setup phase, the system manager generates some system parameters. And some important notations need to be described as shown in Table I.

$Setup(1^{\lambda}) \rightarrow Para$. Based on security parameter λ, first, system manager selects two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with order p. Second, $\mathbb{G}_1$ and $\mathbb{G}_2$ construct a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and g is a generator of $\mathbb{G}_1$. Besides, two cryptographic hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H' : \mathbb{G}_1 \rightarrow Z_p$ are selected. Finally, system parameter $Para = (\mathbb{G}_1, \mathbb{G}_2, g, e, H, H')$ will be published by the system manager on the blockchain.

2) Preprocessing and Storage Phase: During the preprocessing and storage phase, the user divides file F into n blocks and each block is further divided into s sectors. The file F can

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

be defined as $F = \{b_{jk}\}_{1\leqslant j\leqslant n, 1\leqslant k\leqslant s}$. After this, the user will generate copies of the file F and the corresponding tags. We describe the detailed steps as follows:

$KeyGen(\lambda) \rightarrow (pk, sk)$. The user generates the public key $pk := (v = g^x, u = g^{\alpha x}, g, g^{\alpha}, g^{\alpha^2} \ldots, g^{\alpha^{s-1}})$ and the secret key $sk := (x, \alpha)$ where $x, \alpha \in Z_p^*$ are two random numbers.

$RepGen(F) \rightarrow Copy_i$. In decentralized storage, SSPs are highly likely to collude with each other. If the copies of file F are the same, they can share a copy to deceive the user. So we generate different copies of file F by using symmetric encryption techniques such as AES. The diffusion properties of AES can help the user generate the unpredictable ciphertext. We define the ith copy of file F as $Copy_i = \{m_{ijk}\}_{1\leqslant i\leqslant N, 1\leqslant j\leqslant n, 1\leqslant k\leqslant s}$. And $m_{ijk} \in Z_p$ is calculated by $m_{ijk} = E_K(b_{jk}\|i)$.

$TagGen(sk, f_{m_{ij}}(X), F_{id}) \rightarrow \sigma_{ij}$. By using sk, the user calculates tags for each data block as follows:

$$\sigma_{ij} = \left(H\left(F_{id}\|i\|j\right) \cdot g^{f_{m_{ij}}(\alpha)}\right)^x. \tag{1}$$

We define σ_{ij} as the tag of the jth block of the ith copy. The symbol $F_{id} \in Z_p$ is the identity of the File F. And $f_{m_{ij}}$ is defined as a polynomial with s coefficients $\{m_{ijk}\}_{1\leqslant k\leqslant s}$, i.e., $f_{m_{ij}}(X) = \sum_{k=1}^{s} m_{ijk} \cdot X^{k-1}$. Then, the user uploads $(F_i, \{\sigma_{ij}\})$ to SSPi, where SSPi is defined as the ith SSP.

$TagVerfication(pk, m_{ijk}, F_{id}, \sigma_{ij}) \rightarrow 0/1$. To detect whether the user is honest in generating tags, SSPs will use pk to verify each tag by the following equation:

$$e\left(\sigma_{ij}, g\right) = e\left(H\left(F_{id}\|i\|j\right) \cdot g^{f_{m_{ij}}(\alpha)}, v\right). \tag{2}$$

If all tags are verified successfully, the user and SSPs will agree to the store contract. Otherwise, SSPs will refuse the store contract.

3) Auditing Phase: During the auditing phase, the smart contract verifies the integrity of the original file's all copies. We describe this phase in detail as three steps.

$ChalGen(Para, nounce) \rightarrow chal$. To ensure the randomness of the challenge, the nonce of the latest block is selected by smart contract as the random seed. In each audit, the smart contract generates a random c-elements challenge set $Q = \{(a_j, v_j)\}$ and picks two random numbers $r, \gamma \in Z_p$ by using pseudo-random functions. It should be noted that c is the number of challenge blocks, and $a_j \in [1, n]$ and $v_j \in Z_p$ are the indexes and coefficients of challenge blocks, respectively. After that, the smart contract sends $chal = (Q, r, \gamma)$ to each SSP and publishes $chal$ on the blockchain.

$ProofGen(chal, \{f_{m_{ij}}(X)\}, pk) \rightarrow Proof_i$. After receiving the $chal$, SSPi calculates the respective proofs as follows: SSPi aggregates the tags of all challenge blocks and σ_i is calculated by (3).

$$\sigma_i = \left(\prod_{j=1}^{c} \sigma_{ia_j}^{v_j}\right)^{\gamma^{i-1}}. \tag{3}$$

Based on the challenge set Q, SSPi computes the challenge polynomial $f_{M_i}(X) = \sum_{j=1}^{c} v_j \cdot f_{m_{ia_j}}(X)$ and quotient polynomial $h_i(X) = \gamma^{i-1} \cdot \frac{f_{M_i}(X) - f_{M_i}(r)}{X - r}$. And w_i is calculated by (4).

$$w_i = g^{h_i(\alpha)}. \tag{4}$$

To prevent the on-chain transparent proof from leaking some data blocks, SSPi selects a random number $\varepsilon_i \in Z_p$ and calculates s_i with random masking [30] by (5).

$$\begin{cases} s_i' = \gamma^{i-1} \cdot f_{M_i}(r) \\ R_i = v^{\varepsilon_i} = (g^x)^{\varepsilon_i} \\ s_i = s_i' + \varepsilon_i \cdot H'(R_i) \end{cases}. \tag{5}$$

Finally, SSPi publishes $Proof_i = (\sigma_i, w_i, s_i, R_i)$ on the blockchain.

$Verify(pk, \{Proof_i\}, chal) \rightarrow result$. Upon receiving $Proof_i$ from the blockchain, the smart contract aggregates them by (6).

$$\begin{cases} \sigma = \prod_{i=1}^{N} \sigma_i \\ w = \prod_{i=1}^{N} w_i \\ s = \sum_{i=1}^{N} s_i \end{cases}. \tag{6}$$

Then smart contract verifies the integrity of all copies by (7), where $\eta = \prod_{i=1}^{N} R_i^{H'(R_i)}$ and $P = \prod_{i=1}^{N} (\prod_{j=1}^{c} H(F_{id}||i||j)^{v_j})^{\gamma^{i-1}}$.

$$e\left(\sigma \cdot \eta, g\right) \cdot e\left(g^{-s}, v\right) = e(P, v) \cdot e\left(w, u \cdot v^{-r}\right). \tag{7}$$

The smart contract will publish the $result$ to the blockchain to form an audit log $(chal, \{Proof_i\}, result)$. If the (7) holds, output $result = 1$ to indicate acceptance; otherwise output $result = 0$ to indicate rejection and the smart contract will verify the proof of each SSP by the (8), where $\eta_i = R_i^{H'(R_i)}$ and $P_i = (\prod_{j=1}^{c} H(F_{id}||i||j)^{v_j})^{\gamma^{i-1}}$. The SSPs which fail to pass the verification will be penalized and bear the cost of this round of auditing.

$$e\left(\sigma_i \cdot \eta_i, g\right) \cdot e\left(g^{-s_i}, v\right) = e\left(P_i, v\right) \cdot e\left(w_i, u \cdot v^{-r}\right). \tag{8}$$

TABLE II
SUPPLEMENTARY NOTATIONS AND DESCRIPTIONS IN EFFICIENT BATCH AUDITING SCHEME

Notations	Descriptions
Q_l	The challenge set of the lth file
r_l	The evaluation point of the lth file
$\sigma_{l,ij}$	The tag σ_{ij} of the lth file
$F_{l,M_i}(X)$	The challenge polynomial of ith copy of the lth file
$f_{l,m_{ij}}(X)$	The polynomial $f_{m_{ij}}(X)$ of the lth file

C. Efficient Batch Auditing Scheme

To improve the efficiency of processing batch audit tasks, we can adopt this scheme during the auditing phase. And some supplementary important notations need to be described as shown in Table II.

We assume that the integrity of all copies of d files is verified simultaneously and the symbol l stands for the lth file where $1 \leqslant l \leqslant d$. The specific details of the batch auditing phase are as follows:

$ChalGen(Para, nounce) \rightarrow chal$. The challenge is generated in the same way as in the basic scheme. In each audit

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

period, the smart contract sends random challenges $chal = (\{Q_l\}, \{r_l\}, \gamma, z)$ of all files to each SSP. Q_l and r_l denote the random challenge set and random evaluation point of the lth file, respectively. $\gamma, z \in Z_p$ are two random numbers.

$ProofGen(chal, \{f_{l,m_{ij}}(X)\}, pk) \rightarrow Proof_i$. Based on $chal$, SSPi calculates the respective proofs in the following steps. SSPi aggregates the tags of all challenge blocks and σ_i' is calculated by (9).

$$\sigma_i' = \prod_{l=1}^{d} \left(\prod_{\{a_j, v_j\} \in Q_l} \sigma_{l,ia_j}^{v_j} \right)^{\beta_l}. \tag{9}$$

According to the input of the $ProofGen$ algorithm, SSPi computes the challenge polynomial $F_{l,M_i}(X) = \sum_{\{a_j,v_j\}\in Q_l} v_j \cdot f_{l,m_{ia_j}}(X)$and the value $S_{l,i} = F_{l,M_i}(r_l)$ at its evaluation point. It should be noted that $f_{l,m_{ij}}(X)$ is represented as the polynomial of the jth block of the ith copy of the lth file. Furthermore, fix the set $T = \{r_1, \ldots, r_d\}$, the set $S_l = \{r_l\}$ and the polynomial $Z_S := \prod_{z\in S}(X - z)$, some polynomials are calculated by (10) where $\beta_l = \gamma^{l-1} \cdot Z_{T\setminus S_l}(z)$.

$$\begin{cases} F_i(X) = & \sum_{l=1}^{d} \gamma^{l-1} \cdot Z_{T\setminus S_l}(X) \\ & \cdot (F_{l,M_i}(X) - F_{l,M_i}(r_l)) \\ H_i(X) = & \frac{F_i(X)}{Z_T(X)} \\ L_i(X) = & \sum_{l=1}^{d} \beta_l \cdot (F_{l,M_i}(X) - F_{l,M_i}(r_l)) \\ & - Z_T(z) \cdot H_i(X) \end{cases}. \tag{10}$$

After obtaining the above polynomials, the proof value W_i, W_i', E_i is calculated by (11). (Note that $L_i(z) = 0$. Thus $(X - z)$ divides $L_i(X)$ and W_i' can be calculated by the coefficient of $\frac{L_i(X)}{X-z}$ and pk without secret key α.)

$$\begin{cases} W_i = g^{H_i(\alpha)} \\ W_i' = g^{\frac{L_i(\alpha)}{\alpha - z}} \\ E_i = \sum_{l=1}^{d} \beta_l \cdot S_{l,i} \end{cases}. \tag{11}$$

Finally, SSPi publishes $Proof_i = (\sigma_i', W_i, W_i', E_i)$ on the blockchain.

$Verify(pk, \{Proof_i\}, chal) \rightarrow result$. Upon receiving $Proof_i$ from the blockchain, the smart contract aggregates them by (12).

$$\begin{cases} W = & \prod_{i=1}^{N} W_i \\ W' = & \prod_{i=1}^{N} W_i' \\ E = & \sum_{i=1}^{N} E_i \\ \sigma' = & \prod_{i=1}^{N} \sigma_i' \end{cases}. \tag{12}$$

Then smart contract verifies the integrity of all copies of d files by (13), where $\psi = g^{-E} \cdot W^{-Z_T(z)}$ and $\zeta = \prod_{i=1}^{N} \prod_{l=1}^{d} (\prod_{\{v_j\}\in Q_l} H(F_{id}\|i\|j)^{v_j})^{\beta_l}$.

$$e(\sigma', g) \cdot e(\psi, v) = e(\zeta, v) \cdot e\left(W', u \cdot v^{-z}\right). \tag{13}$$

The smart contract will publish the $result$ to the blockchain to form an audit log $(chal, \{Proof_i\}, result)$. If the (13) holds, output $result = 1$ to indicate acceptance; otherwise output $result = 0$ to indicate rejection and the smart contract will verify the proof of each SSP by the (14), where $\psi_i = g^{-E_i} \cdot W_i^{-Z_T(z)}$ and $\zeta_i = \prod_{l=1}^{d} (\prod_{\{v_j\}\in Q_l} H(F_{id}\|i\|j)^{v_j})^{\beta_l}$. The SSPs which fail to pass the verification will be penalized and bear the cost of this round of auditing.

$$e(\sigma_i', g) \cdot e(\psi_i, v) = e(\zeta_i, v) \cdot e\left(W_i', u \cdot v^{-z}\right). \tag{14}$$

V. Security Analysis

In this section, we analyze the security of our scheme according to the following theorems.

A. Correctness

The analysis in this part is that the verification stage will always pass if the proof is correctly computed.

Theorem 3: If the users, the SSPs, and the smart contract are honest in obeying the auditing process of our scheme, then the proof generated by the SSPs can pass the verification.

Proof: Equation (7) is equivalent to the product of (8) from 1 to N and the relationship between (13) and (14) is the same. Thus, we only prove the correctness of (8) and (14).

The correctness of the (8) is derived as follows:

$$\begin{aligned} & e(\sigma_i \cdot \eta_i, g) \cdot e\left(g^{-s_i}, v\right) \\ &= e\left(\sigma_i \cdot R_i^{H'(R_i)}, g\right) \cdot e\left(g^{-s_i}, v\right) \\ &= e(\sigma_i, g) \cdot e\left(g^{\varepsilon_i \cdot H'(R_i)}, g^{x}\right) \cdot e\left(g^{-s_i}, v\right) \\ &= e(P_i, v) \cdot e\left(g^{\gamma^{i-1} \cdot f_{M_i}(\alpha)}, v\right) \cdot e\left(g^{-s_i'}, v\right) \\ &= e(P_i, v) \cdot e\left(g^{\gamma^{i-1} \cdot \frac{f_{M_i}(\alpha) - f_{M_i}(r)}{\alpha - r}}, v^{\alpha - r}\right) \\ &= e(P_i, v) \cdot e\left(g^{h_i(\alpha)}, g^{x \cdot (\alpha - r)}\right) \\ &= e(P_i, v) \cdot e\left(w_i, u \cdot v^{-r}\right). \end{aligned}$$

The correctness of the (14) is derived as follows:

$$\begin{aligned} & e(\sigma_i', g) \cdot e(\psi_i, v) \\ &= e\left(\zeta_i \cdot g^{\sum_{l=1}^{d} \beta_l F_{l,M_i}(\alpha)}, g^{x}\right) \cdot e(g^{-E_i} \cdot W_i^{-Z_T(z)}, v) \\ &= e(\zeta_i, v) \cdot e(g^{\sum_{l=1}^{d} \beta_l \left(F_{l,M_i}(\alpha) - F_{l,M_i}(r_l)\right) - Z_T(z) \cdot H_i(\alpha)}, v) \\ &= e(\zeta_i, v) \cdot e(g^{L_i(\alpha)}, v) \\ &= e(\zeta_i, v) \cdot e(g^{\frac{L_i(\alpha)}{\alpha - z}}, v^{\alpha - z}) \\ &= e(\zeta_i, v) \cdot e(W_i', g^{x \cdot (\alpha - z)}) \\ &= e(\zeta_i, v) \cdot e\left(W_i', u \cdot v^{-z}\right). \end{aligned}$$

B. Soundness

The analysis in this part includes three theorems. Theorem 4 proves the property of polynomial binding. Theorems 5 and 6

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

prove that a single tag is unforgeable and the proposed scheme is soundness, respectively.

Theorem 4: If a probabilistic polynomial time adversary $\mathcal{A}$ can find a polynomial $f_{m'}(X)$ to forge $g^{f_m(\alpha)}$ (i.e., $g^{f_{m'}(\alpha)} = g^{f_m(\alpha)}$ and $f_{m'}(X) \neq f_m(X)$), we can construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the t-SDH problem.

Proof: Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ that can find a polynomial $f_{m'}(X)$ such that $g^{f_{m'}(\alpha)} = g^{f_m(\alpha)}$, where $f_m(X)$ and $f_{m'}(X)$ are known to $\mathcal{A}$. $\mathcal{A}$ can construct another polynomial $f_{m''}(X) = f_m(X) - f_{m'}(X)$ and obtain $g^{f^{m''}(\alpha)} = g^{f_m(\alpha) - f_{m'}(\alpha)} \in \mathbb{G}_1$. Since $f_{m'}(\alpha) = f_m(\alpha)$ and $f_{m''}(\alpha) = 0$, i.e., α is a root of polynomial $f_{m_2}(X)$. By factoring $f_{m''}(X)$, $\mathcal{B}$ can easily find $SK = \alpha$ and solve the t-SDH problem.

Theorem 5: In the random oracle model, if the CDH problem is hard, the tag in our proposed scheme is existentially unforgeable under the selection message attack.

Proof: Suppose that there is a PPT adversary $\mathcal{A}$ who can successfully forge the tag. Then we can construct a simulator $\mathcal{S}$ to solve the CDH problem. Given the CDH instance (g, g^a, g^b), $\mathcal{S}$ outputs g^{ab} as the solution of CDH problem by running the random oracle and $\mathcal{A}$. $\mathcal{S}$ works as follows.

- *Setup:* $\mathcal{S}$ generates the system parameter and controls H as the random oracle. Then $\mathcal{S}$ sets g^a and $(g, g^{\alpha}, \ldots, g^{\alpha^{s-1}})$ as the public key, where the secret key x is equivalent to a.
- *Query:* In this step, $\mathcal{A}$ makes queries by submitting $(F_{id}, i, j, f_{m_{ij}}(X))$ to $\mathcal{S}$. Before receiving queries, $\mathcal{S}$ randomly chooses an integer $k^* \in [1, q]$, where q denotes the number of queries to the random oracle. Then $\mathcal{S}$ keeps a query list and responses to $\mathcal{A}$ with $g^w \in G$, where $w \overset{R}{\leftarrow} Z_p$ is randomly chosen by $\mathcal{S}$. It also responds to queries of the form $H(F_{id}\|i\|j)$ as follows. For each i and j, $\mathcal{S}$ chooses a random value $w_k \overset{R}{\leftarrow} Z_p$, where $1 \leq k \leq N * n$ and programs the random oracle as

$$H\left(F_{id}\|i\|j\right) = \begin{cases} g^{b+w_k}/g^{f_{m_{ij}}(\alpha)} & k = k^* \\ g^{w_k}/g^{f_{m_{ij}}(\alpha)} & \text{otherwise} \end{cases}.$$

  Now $\mathcal{S}$ can computes the valid tag σ_{ij} (i.e., σ_k) as

$$\begin{aligned} \sigma_k &= \left(H\left(F_{id}\|i\|j\right) \cdot g^{f_{m_{ij}}(\alpha)}\right)^a \\ &= \left(g^{w_k}/g^{f_{m_{ij}}(\alpha)} \cdot g^{f_{m_{ij}}(\alpha)}\right)^a \\ &= \left(g^a\right)^{w_k}. \end{aligned}$$

  For a query on $(F_{id}, i, j, f_{m_{ij}}(X))$, if it is the k^*th query in the query list, abort.
- *Forgery:* $\mathcal{A}$ forges a tag σ_{ij^*} that has not been queried. If the query of this tag is not the k^*th query, abort. Otherwise, we can get σ_{ij^*} as

$$\sigma_{ij^*} = \sigma_{k^*} = \left(g^{b+w_{k^*}}\right)^a = g^{ab+aw_{k^*}}.$$

Analysis. Based on the above, $\mathcal{S}$ can get the solution $\sigma_{k^*}/(g^a)^{w_{k^*}} = g^{ab+aw_{k^*}}/g^{aw_{k^*}} = g^{ab}$ for the CDH instance. Suppose $\mathcal{A}$ can forge the tag with (t, q, ε). $\mathcal{S}$ will therefore solve the CDH problem with $(t + O(q), \varepsilon/q)$.

Theorem 6: If the signature scheme used for tags is unforgeable under the CDH problem and the polynomial commitment scheme is knowledge soundness by the q-DLOG assumption, the Proof is unforgeable in our proposed scheme.

Proof: Here, we analyze the second scheme and prove the theorem (i.e., $Proof_i = (\sigma_i', W_i, W_i', E_i)$ is unforgeable) in a series of games. To avoid repetition, we do not provide the proof process for the first scheme, which is similar to the second scheme.

Game 0. Game 0 is the challenge game defined in Section III-C.

Game 1. Game 1 is the same as Game 0, with one difference that the challenger $\mathcal{C}$ keeps a list of all tags ever issued. If the adversary $\mathcal{A}$ submits a tag that is not in the list, C declares failure and aborts.

Game 2. Game 2 is the same as Game 1, with one difference that $\mathcal{C}$ maintains a list of its responses to tag query made by $\mathcal{A}$. Then, $\mathcal{C}$ observes each instance of Verify algorithm. If $\mathcal{A}$ is successful (i.e., Verify algorithm outputs 1) but his aggregate σ_i' is not equal to $\prod_{l=1}^{d}(\prod_{\{a_j,v_j\}\in Q_l} \sigma_{l,ia_j}^{v_j})^{\beta_l}$, $\mathcal{C}$ declares failure and aborts.

To further analyze the difference in success probabilities between Game 1 and Game 2, we will establish some notation. We suppose that the file causing the abort is composed of the coefficient of $f_{m_{ij}}(X)$, and its corresponding tag issued by $\mathcal{A}$ is $\{\sigma_{ij}\}$. Then, assume $\{Q_l\}$ is the query that causes the abort and $\mathcal{A}$'s response to this query is $(\hat{\sigma_i'}, \hat{W}_i, \hat{W_i}', \hat{E}_i)$. Let the expected response obtained from an honest prover be $(\sigma_i', W_i, W_i', E_i)$, where $\sigma_i' = \prod_{l=1}^{d}(\prod_{\{a_j,v_j\}\in Q_l} \sigma_{l,ia_j}^{v_j})^{\beta_l}$, $W_i = g^{H_i(\alpha)}$, $W_i' = g^{\frac{L_i(\alpha)}{\alpha - z}}$ and $E_i = \sum_{l=1}^{d} \beta_l \cdot S_{l,i}$. By the correctness of the proposed scheme, the expected response will satisfy the (15), (16), and (17). Note that (15) is multiplied by (16) and (17), where the correctness of (16) is obvious and the correctness of (17) has been proved in [24].

$$e\left(\sigma_i', g\right) \cdot e\left(g^{-E_i} \cdot W_i^{-Z_T(z)}, v\right) = e(\zeta_i, v) \cdot e\left(W_i', u \cdot v^{-z}\right) \tag{15}$$

$$e\left(\sigma_i', g\right) = e\left(\zeta_i \cdot g^{\sum_{l=1}^{d} \beta_l F_{l,M_i}(\alpha)}, v\right) \tag{16}$$

$$e(g^{\sum_{l=1}^{d} \beta_l F_{l,M_i}(\alpha)} \cdot g^{-E_i} \cdot W_i^{-Z_T(z)}) = e\left(W_i', u \cdot v^{-z}\right). \tag{17}$$

Because C aborts, we know that $\sigma_i' \neq \hat{\sigma_i'}$ and $\hat{\sigma_i'}$ can pass the (16). Thus we can obtain the (18).

$$e\left(\hat{\sigma_i'}, g\right) = e\left(\zeta_i \cdot g^{\sum_{l=1}^{d} \beta_l F_{l,M_i}(\alpha)}, v\right). \tag{18}$$

Now we will show that we can construct a simulator $\mathcal{S}$ to solve the CDH problem if $\mathcal{A}$ casues $\mathcal{C}$ in Game 2 to abort with non-negligible probability. By challenging a single block of a single file with $Q_1 = (a_1, v_1)$, $\mathcal{S}$ can easily forge a single tag as $\sigma_{ia_1} =$

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

TABLE III
FUNCTION COMPARISONS OF AUDITING SCHEMES

Schemes	Blockchain-based	Storage Model	File Copy	Batch Auditing
Scheme [13]	NO	Cloud Storage	Multiple	NO
Scheme [21]	YES	Decentralized Storage	Single	NO
Our scheme	YES	Decentralized Storage	Multiple	YES

$\hat{\sigma}_i'^{1/v_1\beta_1}$. According to Theorem 5, a single tag cannot be forged. Therefore, $\mathcal{S}$ can embed this forged tag into the proof of Theorem 5 to solve the CDH problem.

Game 3. Game 3 is the same as Game 2, with one difference that $\mathcal{C}$ tracks store queries and observes each instance. If $\mathcal{A}$ is successful in any of these instances but at least one of the $(\hat{W}_i, \hat{W}_i{}', \hat{E}_i)$ is not equal to the expected $g^{H_i(\alpha)}$, $g^{\frac{L_i(\alpha)}{\alpha - z}}$ and $\sum_{l=1}^{d} \beta_l \cdot S_{l,i}$. respectively, $\mathcal{C}$ declares failure and aborts.

By the previous notion in Game 2, we can construct a simulator $\mathcal{S}$ to break the knowledge soundness of PCS-MPAP if $\mathcal{A}$ causes $\mathcal{C}$ to abort with non-negligible in Game 3.

$\mathcal{S}$ interacts with $\mathcal{A}$ until the condition specified in the case of Game 3 occurs: $\mathcal{A}$ passes the Verify algorithm with $(\hat{W}_i, \hat{W}_i{}', \hat{E}_i)$ that is different from the expected response (W_i, W_i', E_i). S takes the $(\hat{W}_i, \hat{W}_i{}', \hat{E}_i)$ as input to pass the (17) which is the same as the verification equation of PCS-MPAP.

Therefore, if there is a nonnegligible difference between the adversary's probabilities of success in Game 2 and Game 3, we can construct a simulator $\mathcal{S}$ that uses $\mathcal{A}$ to break the knowledge soundness of PCS-MPAP and further break the q-DLOG assumption.

VI. PERFORMANCE

In this section, we compare the latest work [13], [21]. As far as we know, in the cloud storage, [13] is an efficient multi-copy auditing proposal designed with ID-based cryptography. [21] is quite an advanced auditing scheme in decentralized storage. Their specific function comparison is shown in Table III. Therefore, we choose the above three schemes as comparisons to highlight the advantages of our work from different perspectives.

A. Theoretical Analysis

We analyze the computation overhead and on-chain storage overhead of the compared schemes. We give some definitions in Table IV for analysis. Furthermore, we omit some operations such as hashing, addition, and multiplication on Z_p because of their low and negligible overhead. We assume that a file is encrypted with N copies. Each copy is stored separately on N SSPs. Each copy has n blocks and each block has s sectors. Besides, we assume that the number of files to be audited in a batch is d and the number of challenge blocks is c for each file. Then, we summarize the comparisons of computation overhead and on-chain storage overhead in Tables V, VI and VII, respectively. One point we want to make is that since multiple SSPs can execute the protocol in parallel, we only consider the computation overhead of a single SSP in the TagGen phase and ProofGen phase.

TABLE IV
NOTATIONS AND DESCRIPTIONS IN PERFORMANCE

Notations	Descriptions
N	The number of copies and the number of SSPs
d	The file number of batch auditing
n	The block number of each copy
s	The sector number of each block
c	The number of challenge blocks
$\|Z_p\|$	The size of the element in Z
$\|\mathbb{G}_1\|$	The size of the element in $\mathbb{G}_1$
$\|\mathbb{G}_T\|$	The size of the element in $\mathbb{G}_T$
$\|Q\|$	The size of the challenge set of a file
$\|Copy\|$	The size of a file copy
T_{Mul}	The computation cost of multiplication operation on $\mathbb{G}_1$
T_{Exp}	The computation cost of exponentiation operation on $\mathbb{G}_1$
T_{Pair}	The computation cost of bilinear pairing operation
T_{Exp}	The computation cost of exponentiation operation on $\mathbb{G}_1$

TABLE V
COMPARISONS OF COMPUTATION COSTS DURING THE PREPROCESSING AND STORAGE PHASE

	TagGen	TagVerify
Scheme [13]	$2NnT_{Exp} + 2NnT_{Mul}$	$3nT_{Pair} + nsT_{Mul} + nT_{Exp}$
Scheme [21]	$2NnT_{Exp} + NnT_{Mul}$	$2nT_{Pair} + nsT_{Mul} + nsT_{Exp}$
Our scheme	$2NnT_{Exp} + NnT_{Mul}$	$2nT_{Pair} + nsT_{Mul} + nsT_{Exp}$

The three schemes all adopt the method of dividing a data block into sectors, so the difference between their computation costs is very small, as shown in Table V. The advantage of this method is that it can efficiently improve computing efficiency during the preprocessing and storage phase.

In the distributed multi-copies scenario, we also compared the overhead of the auditing phase. As shown in Table VI, when auditing a single file, scheme [13] is more efficient than scheme [21] and our scheme when generating proofs. During auditing verification, both our scheme and scheme [13] can process multiple copies distributed on different SSPs at one time, while scheme [21] can only verify one copy. However, limited by the large proof size, the auditing protocol of scheme [13] is not suitable for decentralized storage. In the following comparative analysis, we will further explain the reasons. When the scenario is extended to batch audit tasks with multiple files, the computation overhead of scheme [13] and scheme [21] will grow linearly with the number of files, as shown in Table VII. And our scheme can solve the drawbacks in this scenario.

In the preprocessing and storage phase, all three schemes upload the calculated n tags with a file copy to the SSP. Their communication overhead is the same and the size is $n|\mathbb{G}_1| + |Copy|$. While in the audit phase, on-chain storage overhead is equal to communication overhead. So we no longer analyze communication overhead. As shown in Table VIII, we assume that the auditing protocol of scheme [13] is used in decentralized storage and compare it with its on-chain storage overhead. Obviously, the overhead of scheme [13] will be a huge burden due to the setting of the number of sectors s. And with the help of the polynomial commitment technique scheme [21] and our scheme, the proof size is constant. When batch verifying multiple files, our scheme can still maintain a constant overhead of size $3|\mathbb{G}_1| + |Z_p|$.

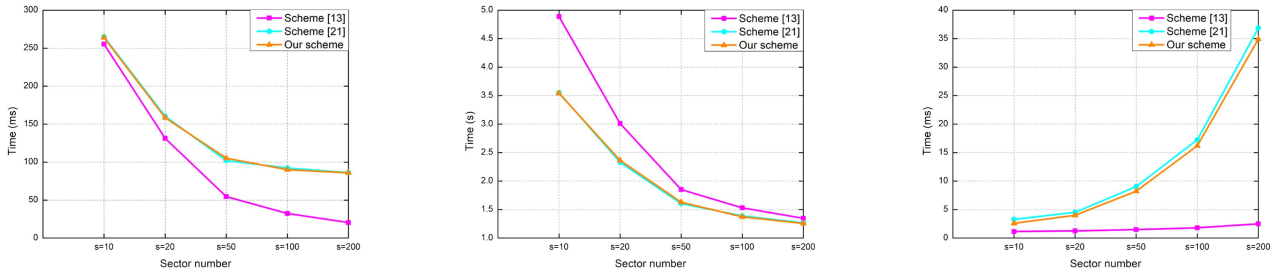
Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
COMPARISONS OF COMPUTATION COSTS DURING THE AUDITING PHASE FOR SINGLE FILE

	ProofGen	Verify
Scheme [13]	$(c-1)T_{Mul}+cT_{Exp}$	$3T_{Pair}+(cN+s+2)T_{Exp}+(cN+s-2)T_{Mul}$
Scheme [21]	$(c+s-3)T_{Mul}+(c+s-1)T_{Exp}$	$4NT_{Pair}+N(c+2)T_{Exp}+NcT_{Mul}$
Our scheme	$(c+s-3)T_{Mul}+(c+s)T_{Exp}$	$4T_{Pair}+(cN+2N+2)T_{Exp}+(cN+3N-2)T_{Mul}$

TABLE VII
COMPARISONS OF COMPUTATION COSTS DURING THE AUDITING PHASE FOR MULTIPLE FILES

	ProofGen	Verify
Scheme [13]	$d(c-1)T_{Mul}+dcT_{Exp}$	$3dT_{Pair}+d(cN+s+3)T_{Exp}+d(cN+s)T_{Mul}$
Scheme [21]	$d(c+s-3)T_{Mul}+d(c+s-1)T_{Exp}$	$4dNT_{Pair}+dN(c+2)T_{Exp}+dNcT_{Mul}$
Our scheme	$(cd+2s-5)T_{Mul}+(dc+d+2s-2)T_{Exp}$	$4T_{Pair}+(Ndc+Nd+3)T_{Exp}+(Ndc+3N-2)T_{Mul}$



(a) TagGen (b) TagVerfication (c) ProofGen

Fig. 2. Comparisons of off-chain computation overhead.

TABLE VIII
COMPARISONS OF STORAGE OVERHEAD ON-CHAIN FOR EACH SSP DURING THE PROOFGEN PHASE

	Single File	Multiple files
Scheme [13]	$\lvert\mathbb{G}_1\rvert+s\lvert Z_p\rvert$	$d\lvert\mathbb{G}_1\rvert+ds\lvert Z_p\rvert$
Scheme [21]	$2\lvert\mathbb{G}_1\rvert+\lvert\mathbb{G}_T\rvert+\lvert Z_p\rvert$	$2d\lvert\mathbb{G}_1\rvert+d\lvert\mathbb{G}_T\rvert+d\lvert Z_p\rvert$
Our scheme	$3\lvert\mathbb{G}_1\rvert+\lvert Z_p\rvert$	$3\lvert\mathbb{G}_1\rvert+\lvert Z_p\rvert$

B. Implementation

To evaluate the performance of the schemes, we conducted a series of experiments to test the on-chain and off-chain overheads of all comparative schemes. We choose to use the curve BN256 supported by the Ethereum precompiled contract for comparative experiments. For on-chain testing, we use the Ganache[1] to simulate the Ethereum blockchain environment in the Alibaba Cloud Server ECS s6. And the solidity language is used to implement smart contract. For off-chain testing, we use golang language to call cloudflare cryptographic library[2] to implement cryptographic operations on curve BN256 ($|Z_p| = |\mathbb{G}_1| = 32$ bytes, $|\mathbb{G}_T| = 64$ bytes, $|\mathbb{G}_T| = 192$ bytes). Furthermore, as we described in the introduction, the decentralized storage model makes better use of personal idle storage resources. Thus, we set up the SSP's test environment on the machines with Intel(R) Core(TM) i5-10500 CPU 3.10 GHz 12.0 GB RAM, Linux and set up the user's test environment on a computer with Intel(R) Core(TM) i5-12400F CPU 2.50 GHz 8.0 GB RAM, Windows. To facilitate testing, the size of each file copy used for the experiment is 320 KB. And it is divided into 10,000 data chunks of size 32B. Notably, all experiments are executed 100 times to obtain an average result.

1) Off-Chain Part Costs: For a clearer exposition, we further analyze the off-chain overhead at each stage. And we fixed the number c of challenge blocks to 10 during a test round of auditing to achieve a fair comparison.

As shown in Fig. 2, the computational overhead of each stage will be affected due to the different settings of the number of sectors s. As the number of sectors s increases, the number of tags that need to be calculated will decrease. Therefore, the computational cost in stage TagGen and TagVerfication is also reduced accordingly. Because of the same tag construction, scheme [21] and our scheme have a similar cost. Correspondingly, in the ProofGen stage, scheme [21] and our scheme require more polynomial operations. Scheme [13] also needs to calculate more proof. In short, dividing the data block into sectors can indeed alleviate the computation overhead at the cost of proof generation time and pre-preparing elements of size $s|\mathbb{G}_1|$ ($\{u_k\}_{1\le k\le s}$ in scheme [13], $\{g^{\alpha^{j-1}}\}_{1\le j\le s}$ in scheme [21] and ours). Thus, the number of sectors needs a reasonable setting.

Next, we will explain why scheme [13] is not suitable for auditing in decentralized storage. According to the Ethereum yellow paper [8], it takes 20,000 units of gas to store a 32-byte

[1]https://trufflesuite.com/ganache/
[2]https://github.com/cloudflare/bn256

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

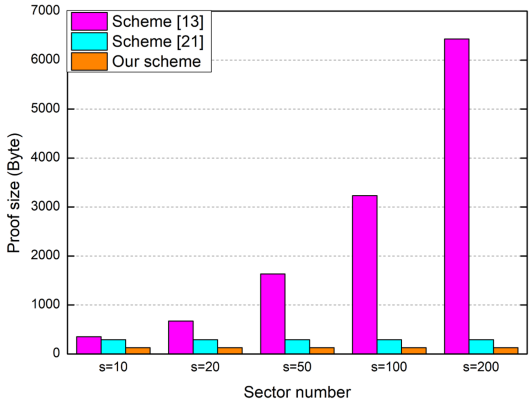


Fig. 3. Comparisons of the proof size for single file.

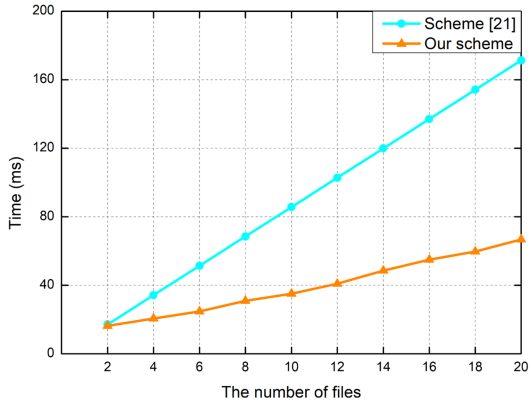


Fig. 5. Comparisons of the single SSP's computation overhead in ProofGen stage for multiple files.

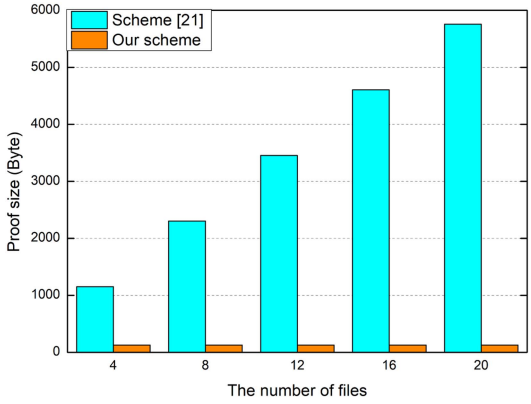


Fig. 4. Comparisons of the proof size for multiple files.

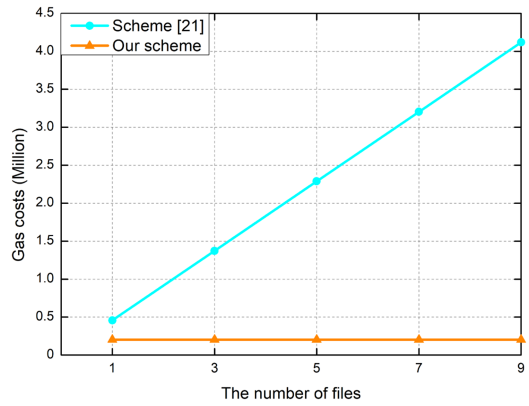


Fig. 6. Comparisons of the single SSP's on-chain storage overhead in ProofGen stage for multiple files.

word on the Ethereum blockchain. So the proof size is critical to an auditing scheme in decentralized storage. Although scheme [13] has high efficiency in cloud storage. Malicious attackers have the opportunity to obtain partial data of users, due to the proof of linear combination of data blocks on the transparent blockchain. Most importantly if it is used directly in a decentralized paradigm, storing huge proof on the blockchain would bring catastrophic on-chain overhead, as shown in Fig. 3. In contrast, scheme [21] and our scheme only generate proof of constant size of 288-byte and 128-byte for each SSP, respectively. Thus, scheme [13] is not suitable for auditing in decentralized storage and we will not discuss scheme [13] in the following comparison.

We set s to 50 to compare the overhead under batch auditing. In the scenario of auditing multiple files at once, PCS-MPAP can open multiple polynomials at distinct points. As shown in Figs. 4 and 5, the overhead of scheme [21] will increase linearly with the number of files. In contrast, our scheme still guarantees constant proof size for each SSP and has lower computation overhead in the face of a large number of files. Consequently, our batch auditing scheme is more efficient than scheme [21], both in terms of computation overhead and proof size.

2) On-Chain Part Costs: We evaluate the on-chain overhead of the scheme based on the Ethereum platform. Gas in Ethereum is the unit for measuring the computational effort required to execute a contract. In brief, the more complex operations to be performed and the more data stored on the blockchain, the more gas costs will be required. Although one challenge we encounter here is the high cost of implementing complex cryptographic primitives on the blockchain due to the limitations of the solidity language. We still use the gas costs as the standard to measure the on-chain resource consumption of each scheme. Our intention is only for a convenient and fair comparison. It should be emphasized that our scheme can be applied to any blockchain platform.

With the number of files increasing, the proof size of scheme [21] to be stored on the blockchain will grow linearly. What is shown in Fig. 6 is consistent with the conclusion of the theoretical analysis. Storing the elements of $3|\mathbb{G}_1| + |Z_p|$ on the blockchain requires 202,905 units of gas. And storing the elements of $2|\mathbb{G}_1| + |\mathbb{G}_T| + |Z_p|$ on the blockchain requires 457,633 units of gas. Obviously, our scheme can maintain a constant gas cost during batch auditing to save a lot of on-chain storage resources.

In the verification stage, we assume that 10 copies of each file are generated and distributed to different SSPs (i.e., $N = 10$). As shown in Fig. 7, when the number of files is 1, our scheme

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

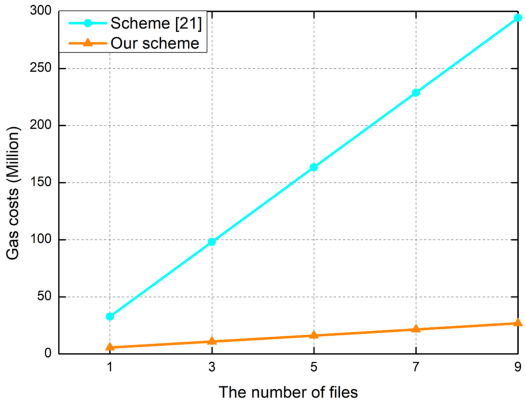


Fig. 7. Comparisons of the on-chain computation overhead in Verify stage for multiple files.

consumes 5,685,545 units of gas to verify its 10 copies at a time. As the number of files increases by 2, that is, the number of verified copies increases by 20, our scheme only needs to consume about 5,000,000 units of gas to perform additional hash operations. But, scheme [21] can only verify a single copy at a time and consume 3,269,246 units of gas for a copy. It will require a lot of on-chain computing resources when there are a lot of audit requests.

In summary, all the experimental results satisfy the previous theoretical analysis. And our scheme achieves the expected purpose.

VII. Conclusion

In this study, we proposed an efficient data integrity auditing scheme for multi-copy in decentralized storage. We first proposed a basic scheme for auditing all distributed file copies simultaneously. Specifically, there was potential to apply polynomial commitment techniques to integrity auditing in the decentralized paradigm. In addition, we further proposed an efficient batch auditing scheme considering a large number of audit requests. It reduced computation overhead while keeping the proof size constant. This will save a lot of on-chain and off-chain resources. Theoretical analysis and experimental results confirmed that our scheme is efficient and suitable for decentralized scenarios. In future work, we aim to further improve the scheme to alleviate the computation overhead of on-chain verification.

Acknowledgments

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this article.

References

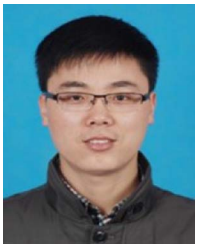
[1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," *Framingham: Int. Data Corporation*, vol. 16, pp. 1–28, 2018.
[2] F. Zafar et al., "A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends," *Comput. Secur.*, vol. 65, pp. 29–49, 2017.
[3] G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.
[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2008, pp. 90–107.
[5] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1421–1432, Third Quarter 2021.
[6] H. Wang and Y. Zhang, "On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 264–267, Jan. 2014.
[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, 2008, Art. no. 21260.
[8] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.
[9] J. Benet and N. Greco, "FileCoin: A decentralized storage network," *Protoc. Labs*, pp. 1–36, 2018.
[10] J. Benet, "IPFS-content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.
[11] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, 2008, pp. 411–420.
[12] A. F. Barsoum and M. A. Hasan, "Provable possession and replication of data over cloud servers," Centre For Applied Cryptographic Research (CACR), University of Waterloo, Report, vol. 32, 2010, Art. no. 2010.
[13] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 356–365, First Quarter 2022.
[14] G. Tian et al., "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3941–3954, Nov./Dec. 2022.
[15] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 923–937, Third Quarter 2021.
[16] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, pp. 1–16, 2019.
[17] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, 2020.
[18] Q. Zhang, D. Sui, J. Cui, C. Gul, and H. Zhong, "Efficient integrity auditing mechanism with secure deduplication for blockchain storage," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2365–2376, Aug. 2023.
[19] 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 Trans. Services Comput.*, vol. 13, no. 2, pp. 289–300, Mar./Apr. 2020.
[20] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2252–2263, Fourth Quarter 2022.
[21] Y. Du, H. Duan, A. Zhou, C. Wang, M. H. Au, and Q. Wang, "Enabling secure and efficient decentralized storage auditing with blockchain," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 30380–3054, Sep./Oct. 2022.
[22] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2010, pp. 177–194.
[23] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," *Cryptol. ePrint Arch.*, Paper 2019/953, 2019. [Online]. Available: https://eprint.iacr.org/2019/953
[24] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, "Efficient polynomial commitment schemes for multiple points and polynomials," *Cryptol. ePrint Arch.*, Paper 2020/081, 2020. [Online]. Available: https://eprint.iacr.org/2020/081
[25] M. Andoni et al., "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable Sustain. Energy Rev.*, vol. 100, pp. 143–174, 2019.
[26] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *Proc. Annu. Int. Cryptol. Conf.*, Springer, 2018, pp. 33–62.
[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2001, pp. 514–532.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.

[28] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2016, pp. 305–326.
[29] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2111–2128.
[30] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE Conf. Comput. Commun.*, 2010, pp. 1–9.

Qingyang Zhang received the BEng and PhD degrees in computer science from Anhui University, in 2021. He is currently an associate professor of the School of Computer Science and Technology, Anhui University. His research interest includes edge computing, computer systems, and security.

Zhiming Zhang is now a research student with the School of Computer Science and Technology, Anhui University. His research focuses on the security of Blockchain.



Jie Cui (Senior Member, IEEE) received the PhD degree from the University of Science and Technology of China, in 2012. He is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has more than 150 scientific publications in reputable journals (e.g., *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Multimedia*), academic books, and international conferences.

Hong Zhong received the PhD degree in computer science from the University of Science and Technology of China, in 2005. She is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking (SDN). She has more than 200 scientific publications in reputable journals (e.g., *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, and *IEEE Transactions on Big Data*), academic books, and international conferences.

Yang Li received the master's degree from the Electronic Engineering Institute, Hefei, China. He is now with Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, Anhui, China. His research interests include vulnerability discovery, network security, and computer science.

Chengjie Gu received the PhD degree from the Nanjing University of Posts and Telecommunications, in 2012. From 2012 to 2017, he was an innovation team leader with the 38th Research Institute of CETC and conducted research and development in the communication and networking sector. Currently, he is a dean of the School of Public Security and Emergency of Anhui University of Science and Technology. He has completed postdoctoral research with the USTC. He is a high-level innovation leader of Anhui province and a cybersecurity expert of Zhejiang province in China. His research interest includes network security and trusted network architecture, etc.

Debiao He received the PhD degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University, and the Shanghai Key Laboratory of Privacy Preserving Computation, MatrixElements Technologies, Shanghai, China. His main research interests include cryptography and information security, in particular, cryptographic protocols. He has published more than 100 research papers in refereed international journals and conferences, such as *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Security and Forensic*, and Usenix Security Symposium. He is the recipient of 2018 IEEE Sysems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 10000 times with Google Scholar. He is in the editorial board of several international journals, such as *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-centric Computing & Information Sciences*.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on December 13,2025 at 05:23:44 UTC from IEEE Xplore. Restrictions apply.