EDCOMA: Enabling Efficient Double Compressed Auditing for Blockchain-Based Decentralized Storage

Haiyang Yu, *Member, IEEE*, Yurun Chen, *Student Member, IEEE*, Zhen Yang, *Member, IEEE*, Yuwen Chen, *Member, IEEE*, and Shui Yu, *Fellow, IEEE*

***Abstract*—Blockchain technology, known for its decentralized and immutable nature, serves as the foundation for various applications. As a prominent application of blockchain, decentralized storage is powered by blockchain technology and is expected to provide a reliable and cost-effective alternative to traditional centralized storage. A major challenge in blockchain-powered decentralized storage is how to guarantee the quality of storage services in decentralized storage nodes (DSNs). Storage auditing can ensure the integrity and security of the stored data. Unfortunately, it incurs additional computational costs for data owners and extra storage overheads for DSNs, which thereby cannot be directly applied to decentralized storage networks consisting of nodes with various computation and storage capacity. In this article, we overcome these problems and minimize additional burdens in storage auditing. We propose EDCOMA, a computation and storage efficient auditing scheme for blockchain-based decentralized storage, in which a double compression method is designed to compress data authenticators using both data and polynomial commitment. To prevent replay attacks on double compression launched by DSNs, we introduce zero knowledge proof and design a compression arithmetic circuit to guarantee the execution of compression operations in DSNs. We analyze the security of EDCOMA under the random oracle model and conduct extensive experiments to evaluate the performance of EDCOMA. Experimental results affirm that EDCOMA outperforms state-of-the-art approaches in both computational and storage efficiency.**

***Index Terms*—Decentralized storage, auditing, double compression, blockchain.**

I. Introduction

BLOCKCHAIN is a revolutionary approach for decentralized networks and was initially introduced in 2008 [1], [2]. It facilitates the establishment of consensus-based peer-to-peer networks, constructing chains of accepted blocks without the need for a central authority or centralized controller. Blockchain has disrupted traditional models of trust and transaction validation, enabling a secure and transparent exchange of information and value across the globe. By providing a trustless and immutable framework, blockchain paves the way for enhanced security and trust in various applications and industries, including finance, supply chain management, healthcare, etc.

One prominent application of blockchain is seen in decentralized storage (DS) [3], where individuals or micro companies can rent their unused hardware storage capacity to others. As a new paradigm of storage, DS leverages blockchain technology to create a storage platform and encourages more micro storage providers to participate in the storage market and share their spare storage resources. According to [3], the total unused storage space of individual computer users exceeds Google's centralized storage capacity, despite the company's significant storage capabilities.

Several DS projects such as Filecoin, Sia, Storj, etc., have been developed in recent years [4], [5], [6]. Compared with centralized cloud storage, DS offers an algorithmic storage market where data owners and storage nodes can participate, allowing data owners to securely store their data by paying the network. On the other hand, the storage nodes contribute their storage resources to the network in exchange for compensation. In this setting, data owners can be all kinds of devices having data outsourcing requirements (e.g., IoT devices, mobile devices, personal computers, etc.), and storage nodes can be either individuals (e.g., PCs, laptops, workstations, etc.) or micro enterprises possessing extra storage resources.

A major challenge in the DS network is to ensure that the data storage service is correctly maintained by the storage node [3], [7]. Since all kinds of storage nodes are allowed to join the DS network, data integrity checking should be mandatory to storage nodes in order to guarantee the quality of storage services. Storage nodes ought to prove that they have maintained unaltered files for a specified period of time, ensuring the integrity and security of the stored data. To achieve this objective, it is crucial for a DS platform to incorporate an auditing mechanism that effectively tracks storage condition of the storage nodes to guarantee data integrity.

Manuscript received 25 October 2023; revised 27 May 2024; accepted 5 June 2024. Date of publication 21 June 2024; date of current version 9 October 2024. This work was supported in part by Beijing Natural Science Foundation under Grant 4232018, in part by the National Natural Science Foundation of China under Grant 62271456, in part by the National Key Research and Development Program of China under Grant 2022YFB3103104, in part by the Major Research Plan of National Natural Science Foundation of China under Grant 92167102, and in part by the R&D Program of Beijing Municipal Education Commission under Grant KM202210005026. *(Corresponding author: Zhen Yang.)*

Haiyang Yu, Yurun Chen, Zhen Yang, and Yuwen Chen are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: yuhaiyang@bjut.edu.cn; S202274122@emails.bjut.edu.cn; yangzhen@bjut.edu.cn; yuwen.chen@bjut.edu.cn).

Shui Yu is with the School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: Shui.Yu@uts.edu.au).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSC.2024.3417337, provided by the authors.

Digital Object Identifier 10.1109/TSC.2024.3417337

Many storage auditing schemes [8], [9], [10], [11], [12], [13], [14], [15] have been proposed to deal with the integrity problem in remote cloud servers. To apply storage auditing to decentralized settings, decentralized storage auditing [16], [17], [18] is also studied recently, which checks decentralized storage nodes (DSNs) periodically to ensure the accurate storage of the data. Since the verification algorithm is normally deployed in the smart contract of blockchain in the DS network, decentralized storage auditing is designed to improve on-chain efficiency to reduce the costs of executing smart contracts. Existing schemes can support block-level auditing, which randomly challenges selected data blocks with high auditing probability. Homomorphic authenticators are also proposed to support aggregatable verification, allowing all authenticators to be verified batchly. Existing schemes can also protect the privacy of challenged data blocks from being exposed to the third-party verifier. However, there are still problems for both storage nodes and data owners in decentralized storage auditing. 1) Compared with notable giant companies in cloud storage, many storage nodes in DS networks are micro companies or even individuals, which can only provide finite storage resources as service. Thus, they are more sensitive to additional storage overhead than cloud storage providers. 2) Since all kinds of data owners are encouraged to join the network and outsource data, many of them could be mobile devices with limited computation resources. As a result, heavy computation overheads from the auditing are unaffordable to them and thereby not suitable to be applied to the DS network. These two problems can greatly hinder more participants from joining and contributing to the DS network, which finally weakens the robustness of the DS platform due to the limited number of nodes in the network. Thus, building a storage and computation both efficient auditing scheme for DS to fit all kinds of storage nodes and data owners of various resources becomes a key problem to be studied.

In DS auditing, additional costs are mainly derived from the storage and calculation of data authenticators [16], [18]. Intuitively, compressing data authenticators is a natural way to reduce the extra storage cost in DS auditing. However, this natural idea incurs challenges in the following ways. First, the compression method may bring computational burdens to data owners. If the compression method is computation-expensive, the data owner will not be willing to compress authenticators for saving the storage of storage nodes. Second, the introduction of the compression method may become the loophole or vulnerability of the auditing. The storage node may cheat by simply providing an incorrect compression result without storing any data. It can also pass the auditing by storing pre-compressed data instead of original data. Therefore, the verifiability of both the compression result and the compression process should be taken into account in the auditing.

To address the aforementioned challenges, we propose EDCOMA, an **E**fficient **D**ouble **COM**pressed **A**uditing scheme for blockchain-based decentralized storage. In our proposed EDCOMA, we provide not only the compression method, but also the compression verification method. We design a double compression method using both the data and polynomial commitment protocols. Furthermore, inspired by zero knowledge proof (ZKP) protocols used in blockchain, we present the verification method for double compression, designing a compression arithmetic circuit using the ZKP protocol. The double compression verification method can verify the execution of the compression process in DS nodes. In our scheme, the basic off-chain construction of EDCOMA is executed off-chain. To extend EDCOMA to support on-chain auditing, the verification in the on-chain construction is executed on-chain by the smart contract. The major contributions can be summarized as follows:

- We propose an efficient double compressed auditing scheme EDCOMA for blockchain-based decentralized storage, including both off-chain and on-chain constructions. The proposed scheme can check data integrity in decentralized storage nodes with both storage and computational efficiency.
- We present a double compression method using both data and polynomial commitment, and design a novel data authenticator based on this method. Double compression not only minimizes the additional storage overhead incurred by authenticators, but achieves lightweight auditing preparation.
- Based on the ZKP protocol, we provide the compression verification method by designing a compression arithmetic circuit in ZKP, which prevents potential replay attacks from decentralized storage nodes. In this way, the proposed method can guarantee the execution of compression and data aggregation in decentralized storage nodes.
- We conduct security analysis of the proposed EDCOMA under the random oracle model [19], [20], which proves both the correctness and soundness of our scheme. We also analyze the auditing probability in EDCOMA. We implement a prototype to evaluate the performance of EDCOMA and conduct extensive experiments. Experimental results demonstrate that EDCOMA incurs lower storage and computational costs of pre-auditing and verification than the state-of-the-art approaches.

The remaining of this article is organized as follows. We give a literature review in Section II. We introduce the preliminaries of this work in Section III. Section IV presents the novel design and concrete construction of our work. The security analysis of EDCOMA is provided in Section V. Section VI deals with the performance evaluation. In the end, Section VII concludes this article.

II. Related Work

A. Decentralized Storage

Decentralized storage (DS) platforms are storage systems based on technologies including blockchain, distributed file system, smart contract, data encryption, etc., where data is stored across multiple nodes in the network instead of being centralized on a single server. IPFS (Inter Planetary File System) [21] is a peer-to-peer distributed file system designed to create a persistent and distributable file system. IPFS stores files in a dispersed manner across multiple nodes in the network to improve data redundancy and availability.

Another decentralized cloud storage platform is Sia [5], which uses blockchain to realize distributed storage of files. Sia encrypts uploaded files end-to-end to ensure security during transmission and storage. In addition, Sia uses smart contracts to manage storage contracts and payments, ensuring that storage providers provide services according to the contracts and receive appropriate rewards. Sia uses Merkle hash tree (MHT) [22] to authenticate user data and store the tree root on the blockchain.

Storj [6] was originally launched in 2014 as a decentralized Ethereum-based DS system, mainly targeting smart contract platforms for financial applications. Storj has introduced a role called satellite, which is responsible for managing various services in the storage network, including data auditing, node management, data distribution, etc. Participants can also provide idle storage space for network platforms to use, while obtaining a certain value return.

B. Storage Auditing

In the past decade, researchers have proposed many research work on cloud storage auditing. To ensure the integrity of data stored in the cloud, Ateniese et al. [23] first proposed the concept of provable data possession (PDP), which allows to check whether remote servers possess the original data without retrieving it. Juels and Kaliski [24] presented another scheme named proof of retrievability (POR), which includes special blocks called sentinels that are randomly embedded in the data for detection purposes. Based on these schemes, a series of subsequent research efforts have been proposed and storage auditing was extended to support various advanced features such as multi-replicas [25], [26], dynamic updates [27], [28], public authentication [29], etc.

As an emerging distributed technology featured with tamper proofing, blockchain is considered as a solution to be used in storage auditing. Wang et al. [30] proposed the concept of non-interactive public provable data possession and designed a blockchain-based data integrity auditing scheme. However, this solution did not support data dynamics and batch auditing. Li et al. [31] designed a scheme to store hash tags of encrypted file blocks on the blockchain and use Merkle hash trees (MHTs) to verify data integrity. Wang et al. [32] integrated the blockchain and built a decentralised auditing framework using blockchain nodes responsible for auditing, thus solving the trust issue of third-party auditors. However, this solution did not consider using blockchain to store data. Miao et al. [33] proposed a blockchain based cloud storage auditing scheme that supports fault location under multi-cloud storage.

In order to check the integrity of blockchain-based decentralized storage, researchers considered investigating DS auditing schemes. Yue et al. [34] proposed a decentralized auditing scheme to store the roots of MHTs in a blockchain, which unfortunately was inefficient in Big Data scenarios. Ateniese et al. [35] proposed an efficient continuous data integrity auditing scheme for DS to save communication costs. However, as each file must be preprocessed and individually verified, their scheme is not suitable for Big Data scenarios. Subsequently, Yu et al. [36] designed a data sampling strategy to tackle this problem, making the scheme applicable to Big Data scenarios. However, both schemes could only support a limited number of auditing. Chen et al. [16] has designed an auditing protocol for DS based on smart contracts, but the cost of this scheme is still high because homomorphic authenticators must be calculated for each data block. Du et al. [18], [37] proposed a DS auditing scheme that introduces polynomial coefficients to construct homomorphic authenticators, thereby reducing the cost of authenticator generation. In addition, they use smart contracts as third-party auditors in the scheme, enabling automated execution and avoiding additional trust costs. However, these schemes still incur inefficiency in pre-auditing and authenticator storage, which impedes the application of storage auditing in DS.

TABLE I
COMPARISON OF DIFFERENT SCHEMES

Properties	Shen [28]	Chen [16]	Du [18]	EDCOMA
Public Audit	√	√	√	√
Probabilistic	√	√	√	√
Storage Type	Cloud	DS	DS	DS
Poly. Comt.	×	×	√	√
Doub. Comp.	×	×	×	√
zk-SNARK Bs.	×	×	×	√

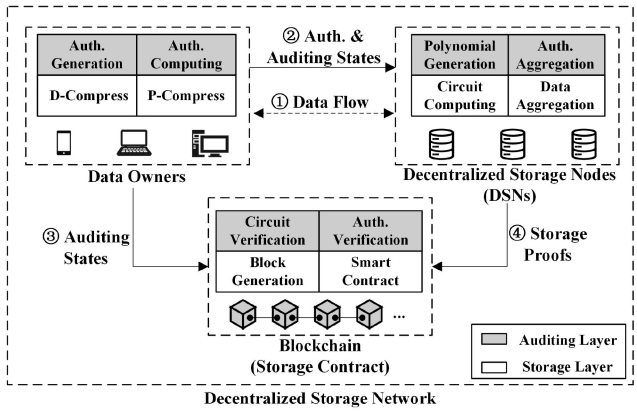


Fig. 1. The system model of EDCOMA, including three parties interacting with each other to perform the auditing task.

A comparison of EDCOMA and existing works is presented in Table I. Existing schemes can support public auditing and probabilistic auditing. However, compared with our proposed EDCOMA, existing works cannot support double compression to reduce the additional storage costs of authenticators, and cannot verify the compression process based on zk-SNARK.

III. MODELS AND PRELIMINARIES

A. System Model

As shown in Fig. 1, we consider three parties in the system model of EDCOMA as explained below.

1) *Data owner:* Data owners outsource local data to multiple decentralized storage nodes in the network and want to ensure the storage correctness of their data. Data owners

are devices with various computational resources, e.g., mobile phones, laptops, PCs, etc. As a consequence, the auditing scheme should not incur heavy loads to data owners. Otherwise, it will discourage massive resource-constrained data owners from joining the DS network, which deviates from the original intention of DS.

2) *Decentralized storage nodes (DSNs):* DSNs provide their spare storage resources as services to data owners. Although they are storage providers in a decentralized storage network, they do not have infinite storage volumes compared with giant cloud storage providers. Thus, they are more sensitive to additional storage costs brought by auditing.
3) *Blockchain:* The blockchain is maintained by data owners and DSNs in the network. It stores transactions between data owners and DSNs in the blocks. The storage contracts (SCs) stored in the blockchain are used for auditing the data in DSNs on behalf of data owners.

Fig. 1 also illustrates the interaction among three parties and the main processes in the auditing layer and the storage layer of each entity. The data owner first pre-processes files by double compressing them in the storage layer, and then generates the authenticators as well as corresponding auditing state in the auditing layer. It then uploads file chunks, authenticators and auditing state to the DSNs. It also passes the auditing state to the storage contract in the blockchain. Each DSN receives files and stores them on its storage servers. The SC on the blockchain challenges the DSN with challenge parameters and the DSN generates the aggregated data by calculating the compression arithmetic circuit based on the ZKP protocol. It also constructs the proof polynomial with aggregated data and aggregates the data authenticators. The DSN forwards all proofs of challenged file chunks to SC. Finally, SC verifies the proofs of circuit as well as authenticators to check whether data are intactly stored and double compressed by the DSN.

B. Definition of EDCOMA

Definition 1: EDCOMA features the following five algorithms namely Setup, Store, Chal, Prove and Verify :

$\mathsf{Setup}(1^{\lambda}) \rightarrow (\varsigma, \delta, pm, k_e, k_v, \Psi)$: The Setup algorithm is responsible for initiating the whole system and is executed by the data owner. On input security parameter λ, this algorithm creates the secret value ς, the secret key δ, public parameters pm, key pairs (k_e, k_v), and compression arithmetic circuit Ψ in ZKP.

$\mathsf{Store}(\varsigma, \delta, r_D, r_P, F) \rightarrow (\{\varrho_i\}, st)$: The Store algorithm is used for compressing the file chunks to generate the corresponding data authenticators and the auditing state. It is run by the data owner. This algorithm takes as input the secret value ς, the secret key δ, data block compression ratio (D-compression ratio) r_D and polynomial compression ratio (P-compression ratio) r_P, and the file F. It outputs a set of data authenticators $\{\varrho_i\}$ and the auditing state st.

$\mathsf{Chal}(F_{info}) \rightarrow \{I_1, I_2, l\}$: The Chal algorithm represents the challenge algorithm and is responsible for generating the challenge parameters. It is executed by the data owner/third party verifier in off-chain auditing scenario, and run by the storage contract in on-chain auditing scenario. Given information of the file F_{info} as input, this algorithm generates the keys (I_1, I_2) and challenge number l.

$\mathsf{Prove}(I_1, I_2, l, r_D, r_P, k_e, \Psi, F, \{\varrho_i\}) \rightarrow \mathsf{Prf}$: The Prove algorithm is run by the decentralized storage node, which generates the proof of storage corresponding to the challenge. Taking as input the secret keys (I_1, I_2), challenge number l, compression ratios (r_D, r_P), proof key k_e, circuit Ψ, the file F and the data authenticators $\{\varrho_i\}$, the Prove algorithm outputs the proof $\mathsf{P}rf$.

$\mathsf{Verify}(I_1, I_2, l, k_v, \mathsf{Prf}, pm) \rightarrow \{SUC/FAIL\}$: The Verify algorithm is responsible for verifying the proof generated by the decentralized storage node. It is run by the data owner/third party verifier in off-chain auditing and is run by the storage contract in on-chain auditing. The Verify algorithm takes as input the secret keys (I_1, I_2), challenge number l, verification key k_v, the proof $\mathsf{P}rf$ and the public parameters pm. This algorithm outputs success SUC if the DSN can pass the verification and outputs failure $FAIL$ otherwise.

C. Security Model

We define the security model of the proposed EDCOMA by following the definition in [38]. We say that EDCOMA is secure if it satisfies correctness, hiding and soundness.

Definition 2: EDCOMA offers correctness if given all entities performing honestly in the scheme, the decentralized storage node always passes the verification.

Definition 3: EDCOMA offers hiding if no adversary has any information of data from the proofs.

Definition 4: EDCOMA offers soundness if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has a non-negligible advantage in the following security game between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup: The challenger $\mathcal{C}$ executes the Setup algorithm and outputs the secret value ς, the secret key δ, and public parameters pm, key pairs (k_e, k_v), and compression arithmetic circuit Ψ. It forwards (pm, k_e, k_v, Ψ) to $\mathcal{A}$, and keeps (ς, δ) private.

Queries: The adversary $\mathcal{A}$ makes a number of queries to the challenger $\mathcal{C}$ in PPT.

1) *Hash query:* The adversary $\mathcal{A}$ adaptively queries hash functions to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ computes hash values and forwards them to the adversary $\mathcal{A}$.
2) *Store query:* The adversary $\mathcal{A}$ adaptively sends queries about the data file F to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ executes the Store algorithm to compute the data authenticator as well as the auditing state and returns them to the adversary $\mathcal{A}$.

Challenge: For any file F on which it previously made a Store query, the adversary $\mathcal{A}$ can execute the protocol with the challenger $\mathcal{C}$. In these protocol executions, the challenger $\mathcal{C}$ plays the role of verifier $\mathcal{V}$ and the adversary $\mathcal{A}$ plays the role of prover, who replaces the prover algorithm $\mathcal{P}$ with any PPT algorithm. When an execution of the protocol completes, the adversary $\mathcal{A}$ is provided with the output of $\mathcal{V}$.

Output: $\mathcal{A}$ outputs description of a cheating prover $\mathcal{P}'$ for a new file not previously appeared in Store Queries.

The adversary $\mathcal{A}$ wins this soundness security game, if the challenger $\mathcal{C}$ accepts the proof with non-negligible probability.

D. Cryptographic Primitives

1) Data Commitment: Commitment serves as an "envelope" that prevents a party from altering the data it has committed to, while also keeping the committed data confidential to others until decommitment. In this paper, a commitment scheme is utilized to compress the data and save the additional storage cost of the storage auditing scheme. A non-interactive commitment scheme is defined as a tuple of three polynomial-time algorithms :

$Setup(1^k)$: On input the security parameter k, this algorithm outputs public parameters including the message space M and the commitment space S.

$C(m)$: The commitment algorithm allows a committer (the one who computes a message to a commitment string) to commit to a message and produce a commitment, which is a fixed-length string, hiding the actual value while binding the committer to that specific message. This algorithm is run by the committer and takes as input a message to be committed $m \in M$ and outputs a commitment string $c \in S$, which is to be publicly published.

$D(c, m)$: The decommitment algorithm allows the receiver to verify the committed value at a later time. This algorithm is run by the receiver and takes as input a commitment string $c \in S$ and a claimed committed message $m \in M$. If $c = C(m)$ then output 1; otherwise output 0.

2) Polynomial Commitment: Let $\beta_0, \beta_1, \ldots, \beta_d \in Z_p$ be a polynomial coefficients and $\alpha \in Z_p$ be a random value, the polynomial commitment on point α with degree d is denoted as $P(\alpha) = \beta_0 + \beta_1 \cdot \alpha + \beta_2 \cdot \alpha^2 + \cdots + \beta_d \cdot \alpha^d \ mod\ p$.

3) Zero Knowledge Proof: Technically, the implementation of zero knowledge proof (ZKP) relies on zk-SNARK [39], [40]. Its key point lies in encoding user-defined computations as quadratic programs. The fundamental process involves initially compiling the program from a high-level language into an arithmetic circuit. Subsequently, this circuit is utilized to construct a quadratic arithmetic program (QAP) consisting of three sets of polynomials $u = \{u_i(x)\}_{i=0}^m, v = \{v_i(x)\}_{i=0}^m, w = \{w_i(x)\}_{i=0}^m$ and a target polynomial $q(x)$, where m is the number of wires in the circuit. Defining polynomial $p(x) = u(x)v(x) - w(x)$, and $q(x)$ divides $p(x)$ iff all wire values are valid assignments for the circuit. The prover constructs $p(x)$ for the proof Φ, and the verifier can verify the correctness by checking if $q(x)$ can divide $p(x)$. The zero-knowledge property can be efficiently incorporated with minimal overhead by introducing three additional random samples and including them as exponents in $u(x)$, $v(x)$, and $w(x)$, respectively.

IV. The Proposed Scheme

In this section, we present the constructions of EDCOMA in detail. First, we present an overview of the proposed scheme. Second, we introduce the major innovations in EDCOMA. Third, we elaborate the detailed construction of EDCOMA for off-chain auditing scenario. Finally, we extend the off-chain auditing scheme to support on-chain auditing, and describe the detailed construction of EDCOMA for on-chain auditing.

TABLE II
Notation Description

Notation	Description
r_P	P-compression ratio
r_D	D-compression ratio
p	Prime order of the group
G_1	Multiplicative cyclic group
F	The stored file
ρ	Public parameters for polynomial commitment
σ	Public values for auxiliary proof
(k_e, k_v)	Key pairs in ZKP
Ψ	Compression arithmetic circuit
ς	Secret value for polynomial commitment
δ	Secret key of data owner
ϱ_i	Data authenticator of the file chunk F_i
st	Auditing state
I_1	Random key for pseudo-random permutation
I_2	Random key for pseudo-random function
l	Challenge number
Ω	Challenge set
η_i	Coefficient in the challenge set
$\pi(\cdot)$	Pseudo-random permutation
$f(\cdot)$	Pseudo-random function
$P_\Psi(\cdot)$	Proof function in ZKP
C_a	Linearly aggregated outputs (commitments)
π	The corresponding proof calculated in ZKP
ω_ϱ	Auxiliary proof
Prf	Proof generated from DSN
$V_\Psi(\cdot)$	Verification function in ZKP
θ	Coefficients of the polynomial

A. Overview

In EDCOMA, we investigate the storage-and-computation efficient storage auditing scheme named EDCOMA. We design the double compression method to greatly save the extra storage and computational costs incurred by data authenticators in storage auditing. Due to the introduction of the commitment protocol in double compression, we further design a compression arithmetic circuit to verify the commitment performed in DSNs by using ZKP. The double compression is performed by the data owner and the circuit is executed by the DSN. A table for notations in our scheme is illustrated in Table II.

B. Innovations

1) Double Compression: In EDCOMA, we design a novel double compression method to construct the data authenticators with compressed files, which saves both the storage and computation cost. The first compression step is data block compression (D-compression), which compresses groups of data blocks in a file chunk to generate coefficients according to the D-compression ratio r_D. D-compression ratio r_D determines the size of each group of data blocks to be input in one data block commitment [41], [42]. The compressed output is the coefficients for the next step. The second step is to use these coefficients to construct a compression polynomial by P-compression ratio r_P [37], [43]. The degree of the compression polynomial is determined by P-compression ratio r_P. Finally, the compression

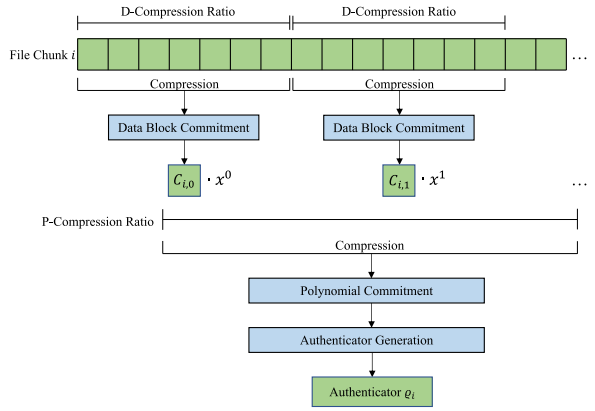


Fig. 2. The detailed design of double compression method, consisting of data block compression (D-compression) and polynomial compression (P-compression).

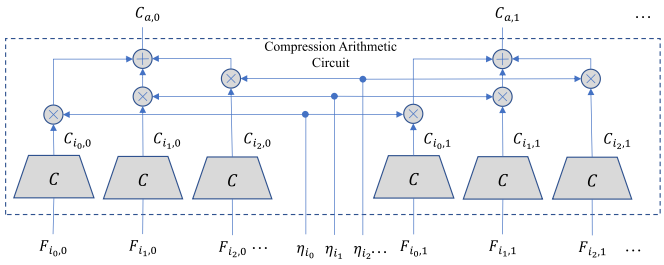


Fig. 3. The detailed design of the compression arithmetic circuit in ZKP. The compression arithmetic circuit includes two logical layers: commitment and linear aggregation layers.

polynomial will be used to perform polynomial commitment on a random point. Both D-compression and P-compression are performed by the data owner to further generate the data authenticator. The data authenticator is calculated based on the double compressed value output by the polynomial commitment. The double compression method is illustrated in Fig. 2.

2) Compression Arithmetic Circuit in ZKP: The aforementioned double compression method is intended to save the storage and computation cost of data authenticators. However, if the compression method itself cannot be verified (especially the first step D-compression), the DSN can cheat the data owner or SC by only storing compressed values instead of data owner's original files. Thus, we adopt ZKP protocols and design a compression arithmetic circuit, which can prove the execution of data block compression in the DSNs. Fig. 3 shows the concrete design of the compression arithmetic circuit, which generally has two logical layers in this circuit. The first layer is the commitment modules for each group of data blocks in the challenge set. These modules take data blocks as input and output the corresponding commitments. The second layer is the linear aggregation of commitments. The linear aggregation is to aggregate the commitments by multiplying the corresponding challenge coeffiecients $\{\eta\}$ generated by in the challenge set. By moving the linear aggregation process into the circuit, not only the output size will be reduced, but the whole compression arithmetic circuit will be resistant to the replay attack. This is because by introducing the randomness of the challenge coefficients $\{\eta\}$ from the linear aggregation, the corresponding proofs of the compression arithmetic circuit will not be constant and will be dynamically changed with the change of randomly generated challenge coefficients. In this way, the DSN who performs the circuit cannot launch replay attacks to the compression verification.

C. Detailed Construction of EDCOMA for Off-Chain Auditing

In this section, we introduce the construction of our proposed EDCOMA for off-chain auditing, which checks the storage via the data owner. The framework consists of two types of entities: decentralized storage nodes (DSNs) and the data owner. In the off-chain auditing, the scheme is performed off-chain and the algorithms of the scheme are executed by the DSNs and the data owner. The DSN runs the Prove algorithm to generate proofs. The data owner sets up the whole systems and then runs the Store, Chal and Verify algorithms to store and verify its data.

Setup: Let $P(x)$ be a polynomial with the degree of $r_P - 1$. g and h denotes two generators of the multiplicative cyclic group G_1 with the prime order p. The data owner generates the secret value ς and the secret key δ. It computes the public parameters $\{\rho_i\}_{i\in[0,r_P-1]}$, where $\rho_i = g^{\varsigma^i}$. It also generates the public values $\{\sigma_i\}_{i\in[0,|F|]}$, where $\sigma_i = h^{\delta^i}$ and $|F|$ is the number of file chunks in file F. It generates the key pairs (k_e, k_v) and the compression arithmetic circuit Ψ for ZKP, and compiles the circuit locally.

Store: The data owner splits each file chunk $F_i \in F$ into $r_P \times r_D$ data blocks $F_i = \{F_{i,0}, F_{i,1}, \ldots, F_{i,r_P\times r_D-1}\}$. It first performs D-compression and compresses each group of data block $\{F_{i,j}\}$ by committing each data block as $C_{i,s} = C(F_{i,j}, F_{i,j+1}, \ldots)$, where $j \in [sr_D, (s+1)r_D - 1]$ and $s \in [0, r_P - 1]$. After D-compression, the data owner performs P-compression and compresses the data by using polynomial commitment. The polynomial of each file chunk is represented as $P_i(x) = C_{i,0} + C_{i,1}x + C_{i,2}x^2 + \cdots + C_{i,r_P-1}x^{r_P-1}$. It computes the polynomial commitment as $P_i(\varsigma)$.

The data owner generates the data authenticator $\varrho_i \leftarrow g^{P_i(\varsigma)}$ and computes the auditing state as $st \leftarrow h^{\prod_{i\in F}(C_G(\varrho_i)+\delta)}$, where $C_G : G_1 \rightarrow Z_p^*$ is a commitment that commits an element in G_1 to a value in Z_p^*.

The data owner transmits all file chunks $\{F_i\}$, authenticators $\{\varrho_i\}$ and the auditing state st to the DSN.

Chal. The data owner sends the challenge parameters (I_1, I_2, l) to the DSN, where I_1 and I_2 are random keys and l is the challenge number (number of challenged file chunks).

Prove: Given the challenge parameters, the DSN computes the challenge set $\Omega = \{(i, \eta_i)\}$, where $i \leftarrow \pi_{I_1}(k)$ is the index of the challenged chunk for $k \in [1, l]$, and $\eta_i \leftarrow f_{I_2}(k)$ is the coefficient for proof generation. $\pi(\cdot)$ and $f(\cdot)$ denote the pseudo-random permutation and the pseudo-random function, respectively.

According to the challenge set Ω, the DSN constructs the proof polynomial $P(x)$. The DSN takes as input the data blocks of the challenged file chunks $\{F_{i,j}\}$, and calculates the compression arithmetic circuit Ψ in ZKP as described in Section IV-B2.

Algorithm 1: Prove: The Storage Proof Generation Algorithm.

Input: challenge parameters (I_1, I_2, l), the public values $\{\sigma_i\}$, file chunks $\{F_i\}$, compression ratios (r_D, r_P), proof key k_e, circuit Ψ and data authenticators $\{\varrho_i\}$
Output: proof Prf

for $k = 1$ *to* l **do**
  Calculate random index $i \leftarrow \pi_{I_1}(k)$
  Calculate coefficient $\eta_i \leftarrow f_{I_2}(k)$
end
Generate proof of compression arithmetic circuit Ψ by $\pi \leftarrow P_\Psi(k_e, \Psi, \{F_{i,j}\}_{i\in\Omega}, C_a, \{\eta_i\}_{i\in\Omega})$
Build the proof polynomial $P(x) = \sum_{i\in\Omega}\eta_i C_{i,0} + \sum_{i\in\Omega}\eta_i C_{i,1}\cdot x + \sum_{i\in\Omega}\eta_i C_{i,2}\cdot x^2 + \cdots + \sum_{i\in\Omega}\eta_i C_{i,r_P-1}\cdot x^{r_P-1}$
Generate coefficients $\{\theta_0, \theta_1, \cdots\}$ according to the polynomial $\prod_{i\in F\setminus\Omega}(h_{\varrho i}+\delta) = \sum_{i=0}^{|F|-|\Omega|}\theta_i\cdot\delta^i$
Generate the auxiliary proof for challenged data authenticator $\omega_\varrho = \omega_{\{\varrho_i\}_{i\in\Omega}} = \prod_{i=0}^{|F|-|\Omega|}\sigma_i^{\theta_i}$
return *the proof* Prf $= (\varrho, \omega_\varrho, C_a, \pi)$

Algorithm 2: Verify: The Proof Verification Algorithm.

Input: the challenge parameters (I_1, I_2, l), the proof Prf , auxiliary proof ω_ϱ, verification key k_v, and the public parameters $\{\rho_i\}$
Output: verification result Res

for $k = 1$ *to* l **do**
  Calculate random index $i \leftarrow \pi_{I_1}(k)$
  Calculate challenge state $\eta_i \leftarrow f_{I_2}(k)$
end
Check the proof π by $V_\Psi(k_v, C_a, \pi)$
if $V_\Psi(k_v, C_a, \pi) == false$ **then**
  return Res $= FAIL$
end
Check the auxiliary proof by $e(st, h) \stackrel{?}{=} e(\omega_\varrho\ , h^{\prod_{i\in\Omega}(h_{\varrho i}+\delta)})$
Calculate $\Phi_P = g^{P(\varsigma)} = \prod_{i=0}^{r_P-1}\rho_i^{C_{a,i}}$
Check $\Phi_P \stackrel{?}{=} \prod_{i\in\Omega}\varrho_i^{\eta_i}$ and output the result Res
return Res

The output of the circuit is C_a, where $C_a = \{C_{a,j}\}_{j\in[0,r_P-1]}$ are the linearly aggregated outputs (commitments). π is the corresponding proof of arithmetic circuit calculation of ZKP.

It generates the auxiliary proof ω_ϱ as

$$\omega_\varrho = h^{\prod_{i\in F\setminus\Omega}(h_{\varrho i}+\delta)} = \prod_{i=0}^{|F|-|\Omega|}\sigma_i^{\theta_i}, \tag{1}$$

for all data authenticators of the challenged file chunks $\varrho = \{\varrho_i\}_{i\in\Omega}$, where $h_{\varrho_i} = C(\varrho_i)$. $\{\theta_0, \theta_1, \ldots\}$ are the coefficients of the polynomial $\prod_{i\in F\setminus\Omega}(h_{\varrho i}+\delta) = \sum_{i=0}^{|F|-|\Omega|}\theta_i\cdot\delta^i$. In the end, it sends all the proofs Prf $= (\varrho, \omega_\varrho, C_a, \pi)$ to the data owner. The concrete proof procedures are summarized and illustrated in Algorithm 1.

Verify: After data owner receives the proofs from the DSN, it first retrieves the challenge set $\Omega = \{(i, \eta_i)\}$ using the same method in the Prove algorithm.

It examines the calculation of commitments of the challenged blocks, by verifying the corresponding proof of arithmetic circuit using $V_\Psi(k_v, C_a, \pi)$. It verifies the auxiliary proof of data authenticators by checking the pairings [44]

$$e(st, h) = e\left(\omega_\varrho, h^{\prod_{i\in\Omega}(h_{\varrho i}+\delta)}\right), \tag{2}$$

where $h_{\varrho_i} = C_G(\varrho_i)$.If (2) does not hold, it aborts.

The data owner then generates $\Phi_P = g^{P(\varsigma)} = \prod_{i=0}^{r_P-1}\rho_i^{C_{a,i}}$. It finally performs the following equation for proof verification

$$\Phi_P = \prod_{i\in\Omega}\varrho_i^{\eta_i}. \tag{3}$$

The algorithm reports success if the equation holds. Otherwise, it reports failure and aborts. The concrete verification procedure is illustrated in Algorithm 2.

D. Detailed Construction of EDCOMA for On-Chain Auditing

In this section, we elaborate the construction of the proposed EDCOMA for on-chain auditing. The framework contains three types of entities: DSNs, storage contract (SC), and the data owner. On-chain auditing of EDCOMA is performed partially on-chain since the SC replaces the data owner to verify the proofs of the DSN. In this construction, the DSN runs Prove algorithm while the verifier is changed from the data owner to SC, which executes the Chal and Verify algorithms in the smart contract. The Chal algorithm is similar to the one in off-chain auditing. The major difference lies in the Verify algorithm since secret values cannot be publicly stored in a smart contract. Otherwise the values will also be accessed by all nodes in the blockchain network.

Since the secret key δ cannot be publicly used, it cannot be stored in SC. Thus, in on-chain auditing, $h^{\prod_{i\in\Omega}(h_{\varrho i}+\delta)}$ in (2) cannot be directly calculated. As a result, we use public values $\{\sigma_i\}$ to generate $h^{\prod_{i\in\Omega}(h_{\varrho i}+\delta)}$ without the knowledge of δ. Therefore, after verifying the proof of ZKP, the SC calculates $h_{\varrho_i} = C_G(\varrho_i)$ for $i \in \Omega$. The $\prod_{i\in\Omega}(h_{\varrho i}+\delta)$ is represented as a polynomial $\epsilon_0 + \epsilon_1\cdot\delta + \cdots + \epsilon_{|\Omega|}\cdot\delta^{|\Omega|} = \epsilon_0 + \epsilon_1\cdot\delta + \cdots + \epsilon_l\cdot\delta^l$. It then calculates $H_\Omega = \prod_{i=0}^{|\Omega|}\sigma_i^{\epsilon_i} = \prod_{i=0}^{l}\sigma_i^{\epsilon_i}$. Finally, SC checks the auxiliary proof by $e(st, h) \stackrel{?}{=} e(\omega_\varrho\ , H_\Omega)$. The rest operations are similar to the ones in off-chain auditing.

We now present the design and structure of the storage contract (SC) in detail. We design SC of our scheme in public blockchain. We select the smart contract of Ethereum platform to demonstrate the structure of SC, and evaluate the performance of SC in Section VI-D. As shown in the Smart Contract, SC first defines variables and then defines the Verify function for validating the proofs of DSN. When passing parameters into the Verify function, a boolean result is returned, which returns true to represent success and false to represent failure. The payable keyword indicates that the function can receive or send ETH. The Verify function performs verification of zero knowledge proof, pairing check for auxiliary proof, etc. Finally, SC pays ETH to the DSN that calls the contract. Auxiliary functions used in the Verify function of SC are also defined and implemented.

Smart Contract: The Structure of SC in On-Chain Auditing.

```
Contract Storage_Contract {
  // Define variables
  ...
  // The Verify function used to
  validate proofs of DSN
  function Verify
  (I_1, I_2, l, Prf, k_v, omega_Varrho, rho, sigma)
  public payable returns (bool) {
  // Calculate the random index i and
  coefficient η_i.
  for (uint k = 1; k<l; k++) {
     index[k] = PRP(I_1, k);
     eta[k] = PRF(I_2, k);
  }
  // Check the proof of ZKP
  require (Verify_Psi(k_v, Ca, Pi) == TRUE,
  "Verification fails");
  ...
  // If the pairing results are not
  equal, verification fails
  Element leftPairing = Pairing(st, h);
  Element rightPairing =
  Pairing(omega_Varrho, H_Omega);
  require (leftPairing == rightPairing,
  "Verification fails");
  ...
  // pay ETH to storage node
  recipient.transfer(paymentAmount);
  return TRUE;
  }
  // Auxiliary functions used for the
  function Verify
  function Pow(···){···}
  function Verify_Psi(···){···}
  ...
}
```

V. Security Analysis

In this section, we provide the security analysis of EDCOMA from two aspects, namely correctness and soundness. We also analyze the auditing probability of the random challenge strategy in EDCOMA.

Theorem 1: If the proposed EDCOMA is honestly executed by all entities in the scheme, the Verify*algorithm will always output SUC.*

The proof of Theorem 1 is given in Appendix A, available online.

Definition 5. t-Strong Diffie-Hellman (t-SDH) Assumption: Let $a \leftarrow Z_p^*$ be a random value and h be the generator of a cyclic group G_1 of prime order p. Given input as a $(t+1)-$tuple $(h, h^a, h^{a^2}, \ldots, h^{a^t}) \in G_1$. For any probabilistic polynomial time adversary, the probability $Pr[(b, h^{\frac{1}{a+b}}) \leftarrow A(h, h^a, h^{a^2}, \ldots, h^{a^t})]$ is negligible for any value of $b \in Z_p^* \backslash -a$.

Theorem 2: Given the t-SDH assumption, the security of the commitment scheme and the zero-knowledge proof protocol, the proofs will be accepted if and only if they are honestly generated by the DSN with an overwhelming probability.

Proof: Suppose that a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can successfully pass the verification with its forged proofs. $(\varrho, \omega_\varrho, C_a, \pi)$ denotes the expected response proofs from an honest prover and $(\varrho', \omega'_{\varrho'}, C'_a, \pi')$ denotes the forged response proof from the adversary $\mathcal{A}$, where $(\varrho, \omega_\varrho, C_a, \pi) \neq (\varrho', \omega'_{\varrho'}, C'_a, \pi')$. We now use a series of cases to analyze the forged parts in the proofs.

Case 1: Assume that the adversary $\mathcal{A}$ provides the forged data authenticators ϱ' and valid auxiliary proof $\omega'_{\varrho'}$. We assume that $\Lambda' = \{h'_{\varrho_0}, h'_{\varrho_1}, \ldots, h'_{\varrho_n}\} \nsubseteq \Lambda = \{h_{\varrho_i}\}_{i\in\Omega}$, where $h'_{\varrho_j} \notin \Lambda$ for some $j \in [0, n]$. The auxiliary proof of ϱ can also be regarded as the proof of Λ, i.e., $\omega'_{\Lambda'} = \omega'_{\varrho'}$. We show a construction that can break the t-SDH assumption. Suppose a tuple challenge $(h, h^\delta, \ldots, h^{\delta^t})$ is given, where $\delta \in Z_p^*$, we show that the adversary $\mathcal{A}$ can compute $(x, h^{1/(\delta+x)})$, where $x \in Z_p$ with non-negligible probability. The adversary $\mathcal{A}$ can output $(h'_{\varrho_0}, h'_{\varrho_1}, \ldots, h'_{\varrho_n})\backslash(-\delta \cup \Lambda)$ and $\omega'_{\Lambda'} \in G_1$ such that

$$\omega'^{\prod_{i\in[0,n]}(h'_{\varrho_i}+\delta)}_{\Lambda'} = h^{\prod_{i\in F}(h_{\varrho_i}+\delta)}, \tag{4}$$

according to (2). From this equation and the tuple challenge, $h^{(1/h'_{\varrho_j}+\delta)}$ can be computed for the $h'_{\varrho_j} \in \Lambda'$, and hence the t-SDH assumption is broken. The adversary can find the solution $(x, h^{1/(x+\delta)})$, where $x = h'_{\varrho_j}$.

As a result, both the have values and auxiliary proofs should be equal, i.e., $\Lambda' = \{h'_{\varrho_i}\}_{i\in\Omega} = \{h_{\varrho_i}\}_{i\in\Omega} = \Lambda$ and $\omega_\varrho = \omega'_{\Lambda'} = \omega_\Lambda = \omega'_{\varrho'}$. Due to the collision-resistance [45] (it is computationally infeasible to find two inputs that map to the same value) of the commitment scheme, we can infer that $\varrho' = \varrho$, either.

Case 2: Assume that the adversary $\mathcal{A}$ forges the aggregated commitment $C'_a \neq C_a$. Since both proofs are valid, we know that both proofs satisfy the (3)

$$\prod_{i\in\Omega} \varrho_i^{v_i} = \Phi_P \tag{5}$$

$$\prod_{i\in\Omega} \varrho_i'^{v_i} = \Phi'_P. \tag{6}$$

Divide (6) by (5), we have

$$\prod_{i\in\Omega} (\varrho'_i/\varrho_i)^{v_i} = \Phi'_P/\Phi_P = g^{(P'(\varsigma)-P(\varsigma))}. \tag{7}$$

Since we have $\varrho'_i = \varrho_i$ in Case 1, we can rewrite (7) as

$$1 = g^{(P'(\varsigma)-P(\varsigma))}. \tag{8}$$

Therefore, we can obtain that $P'(\varsigma) = P(\varsigma)$.

We know that $P(\varsigma) = C_{a,0} + C_{a,1}\cdot\varsigma + C_{a,2}\cdot\varsigma^2 + \cdots + C_{a,r_P-1}\cdot\varsigma^{r_P-1}$. Meanwhile, $P'(\varsigma) = C'_{a,0} + C'_{a,1}\cdot\varsigma + C'_{a,2}\cdot\varsigma^2 + \cdots + C'_{a,r_P-1}\cdot\varsigma^{r_P-1}$. ς is secret to the adversary $\mathcal{A}$, and $\{C'_{a,i}\}$ are random values queried from random oracle. Therefore, due to the computationally binding in polynomial commitment [43], the probability of finding a $P'(x)$ such that $P'(\varsigma) - P(\varsigma) = 0$ is negligible. Therefore, we get $C_a = C'_a$.

Case 3: Assume that the adversary $\mathcal{A}$ forges the proofs of ZKP, i.e., $\pi' \neq \pi$. Due to the security of ZKP, we can infer that the verification algorithm will output $true$ iff the adversary $\mathcal{A}$ honestly executes the Ψ while using private input $\{F'_{i,j}\}_{i\in\Omega}$ or outputs C'_a different from original ones. If the inputs are different $\{F'_{i,j}\}_{i\in\Omega} \neq \{F_{i,j}\}_{i\in\Omega}$ and the outputs are equal $C'_a = C_a$, this means the adversary can guess a set of commitment inputs $\{F'_{i,j}\}_{i\in\Omega}$ that can be committed and then linearly aggregated to generate $C'_a = C_a$. This contradicts to the hiding/one-way property of the commitment scheme, which thereby is negligible in probability.

Thus, both the inputs and the outputs are different, which means $C'_a \neq C_a$ should be satisfied if $\pi' \neq \pi$. However, we know that $C'_a = C_a$ in Case 2. Thus, we can conclude that the proof $\pi' = \pi$. □

Theorem 3: Given returned proofs from the DSN, the adversary cannot derive any file content hidden in the proofs:

The proof of Theorem 3 is given in Appendix B, available online.

Theorem 4: Let r_D be the data block compression ratio (D-compression ratio), r_P be the polynomial compression ratio (P-compression ratio), and l be the challenge number. Given any single numbers r_D, r_P and l, the auditing probability of EDCOMA is greater than $1-(1-\varepsilon)^{lr_Dr_P}$, where ε denotes the proportion of corrupted data blocks.

The proof of Theorem 4 is given in Appendix C, available online.

VI. Evaluation

In this section, we develop a prototype of EDCOMA and evaluate its communication and computation performance by conducting extensive experiments that compare our approach to state-of-the-art schemes. In addition, we analyze the performance of proposed double compression method in EDCOMA. Finally, we evaluate gas costs of on-chain auditing of SC.

A. Communication Overhead

In this section, we assess the communication cost of EDCOMA. As shown in Table I, [18] shares most properties with EDCOMA, which not only audits decentralized storage, but uses polynomial commitment to compress file chunks. Therefore, we evaluate the communication performance of EDCOMA by comparing it with this state-of-the-art approach [18]. We define $|F|$ and l as the file size and challenged chunks, respectively. $|p|$ denotes the size of the group order and $|G_1|$ to denote the size of the base field.

In the Store algorithm of EDCOMA, the data owner transfers all data authenticators $\{\varrho_i\}$, file chunks $\{F_i\}$, and an auditing state st to the DSN, resulting in the communication cost of $(\frac{|F|}{r_Pr_D|p|}+1)|G_1|+|F|$. In the Chal algorithm, the verifier transfers challenge parameters (I_1, I_2, l), incurring the cost of $2|p|+\log_2 \tilde{}l$. Upon generating proofs, the DSN transmits all proofs $\mathsf{Prf} = (\varrho, \omega_\varrho, C_a, \pi)$ to the verifier. As a result, the communication overhead of the Prove algorithm is $r_P|p|+(l+4)|G_1|$.

TABLE III
Communication Overheads of [18] and the Proposed Scheme

Algorithm	EDCOMA	[18]
Store	$(\frac{\lvert F\rvert}{r_Pr_D\lvert p\rvert}+1)\lvert G_1\rvert+\lvert F\rvert$	$\frac{\lvert F\rvert}{r_P\lvert p\rvert}\lvert G_1\rvert+\lvert F\rvert$
Chal	$2\lvert p\rvert+\log_2 l$	$3\lvert p\rvert$
Prove	$r_P\lvert p\rvert+(l+4)\lvert G_1\rvert$	$\lvert p\rvert+2\lvert G_1\rvert$

Table III presents a theoretical comparison of communication overheads between the state-of-the-art scheme [18] and EDCOMA. Furthermore, to better demonstrate the practical communication overheads of EDCOMA, we also conduct additional experiments in Fig. 4. Fig. 4(a) shows the communication overheads in the Store algorithm, and Fig. 4(b) presents the additional communication costs in the algorithm excluding the file, versus different number of chunks used in [18]. It can be observed that [18] has more communication overheads than EDCOMA in the Store algorithm. In addition, EDCOMA holds a significant advantage in minimizing additional communication costs. This is because by adopting double compression, EDCOMA has fewer file chunks and authenticators. As shown in Fig. 4(c), the communication overheads of both schemes remain constant, since the communication overhead of the Chal algorithm is determined by the transmission of challenge parameters, and both [18] and EDCOMA transfer only random keys with limited overhead. However, it can be observed that EDCOMA incurs lower communication cost than [18]. The reason is that [18] transfers a random value in Z_p^* to the DSN for polynomial commitment, while EDCOMA only transfers a smaller number l (the number of challenged chunks) to the DSN. Fig. 4(d) illustrates that [18] has lower communication costs of the Prove algorithm compared to EDCOMA. However, since EDCOMA adopts a random challenge strategy, it can guarantee a high auditing probability with a constant number of challenged file chunks. As a result, the overhead of EDCOMA can also remain constant with a high auditing probability. The auditing probability will be further discussed in Section VI-C.

B. Computation Overhead

We implemented a prototype of EDCOMA in Golang and conducted experiments using a Desktop PC, with Ubuntu 20.04 LTS, AMD Ryzen 7 5800H CPU@3.20 GHz and 16 GB memory. We utilized Pairing-Based Cryptography (PBC) library Nik-U/pbc in Go. The ZKP protocol is implemented based on gnark library [46]. We use type A pairing parameters in our scheme, in which the length of group order $|p| = 160$bits and the length of the base field $|G_1| = 512$bits.

We present the computational performance comparison of existing approaches [16], [18], [28], and our EDCOMA on the same environment aforementioned. Among these schemes, [28] is a classic auditing scheme for cloud storage, [16] and [18] are auditing schemes for decentralized storage without double compression. [18] also uses polynomial commitment to improve the efficiency in data owner. We compare with all of them

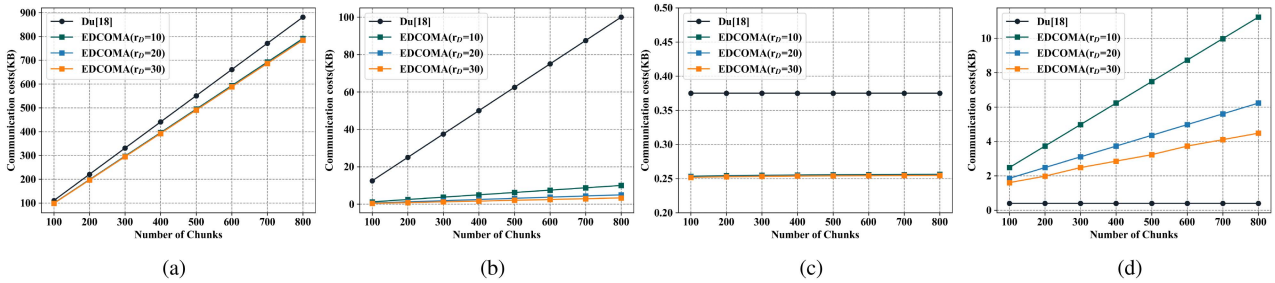


Fig. 4. Comparison of the communication cost in different algorithms. (a) Communication cost of the Store algorithm. (b) Additional communication cost of the Store algorithm. (c) Communication cost of the Chal algorithm. (d) Communication cost of the Prove algorithm.

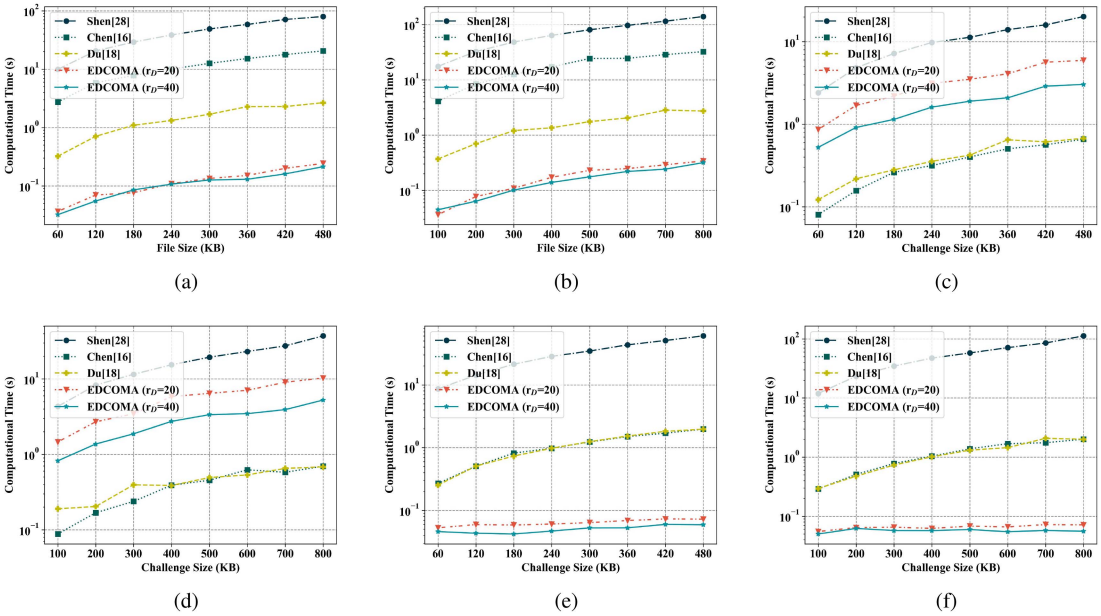


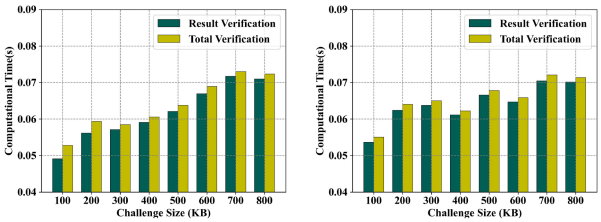
Fig. 5. Comparison of the computational time of [16], [18], [28] and EDCOMA. (a) Computational time of the Store algorithm with $r_P = 30$. (b) Computational time of the Store algorithm with $r_P = 50$. (c) Computational time of the Prove algorithm with $r_P = 30$. (d) Computational time of the Prove algorithm with $r_P = 50$. (e) Computational time of the Verify algorithm with $r_P = 30$. (f) Computational time of the Verify algorithm with $r_P = 50$.

to emphasize the advantage of EDCOMA in computational performance.

In Fig. 5, we present the performance comparison of the Store algorithm's of three approaches [16], [18], [28], and our EDCOMA of two different D-compression ratios ($r_D = 20$ and $r_D = 40$). Specifically, in Fig. 5(a) and (b), we show the performance of all schemes in terms of different file size and P-compression ratio r_P. We can see that the computational time of all schemes increases with the file size. However, our EDCOMA consistently achieves the lowest computational cost in the Store algorithm, regardless of whether $r_P = 30$ or $r_P = 50$. This is because we use the double compression method to reduce the number of data authenticators compared to existing schemes, which thereby also minimizes the computational overheads of data authenticators in the data owner. In addition, we can also find that EDCOMA with $r_P = 50$ has higher efficiency than EDCOMA with $r_P = 30$, which will be further explored in Section VI-C.

The computational overheads of the Prove algorithm are illustrated in Fig. 5(c) and (d). It can be seen that [28] incurs the most computational costs, while [16] and [18] have similar computational overheads in this algorithm. Whether P-compression ratio is 30 or 50, EDCOMA spends more computational time on Prove than [16] and [18], but less than [28]. The main reason is that by introducing the double compression method to save storage and computational costs of data authenticators, we need to utilize ZKP to examine the compression in the DSN during auditing, which can prevent the potential replay attacks from the DSN.

We evaluate the performance of the Verify algorithm among four schemes in Fig. 5(e) and (f). As shown in Fig. 5(e), the computational time of the Verify algorithm of all schemes increases as the challenge size (the size of challenged files) increases. It is easy to see that three schemes including [16], [18] and EDCOMA all have limited computational overheads, and EDCOMA incurs the minimum cost among the three schemes. The reason of the superiority of EDCOMA in the Verify algorithm is that EDCOMA can compress more data blocks into a data authenticator, thereby reducing the challenge number in probabilistic verification. Furthermore, the verification process



(a) $r_p = 30$ (b) $r_p = 50$

Fig. 6. Computational time of verifying compression results in EDCOMA.

is also optimized and simplified. Similar pattern and reason of the performance in the Verify algorithm are also demonstrated in Fig. 5(f).

To evaluate the performance and complexity of compression result verification in the Verify algorithm, we show experiment results in Fig. 6. We can see that the computational time increases as the challenge size increases, whether the ratio $r_p = 30$ or $r_p = 50$. This is majorly because the complexity of compression result verification contains $r_p + l$ exponentiation operations and $r_p + l - 2$ multiplication operations, where l is the number of challenged chunks. It also includes one pairing operation for auxiliary proof verification.

In Prove algorithm, compression arithmetic circuit is employed to ensure that the DSN executes the compression correctly, which also results in more computational time for circuit proof. To mitigate this issue, optimization can be considered by using the distributed zk-SNARK (DIZK [47]) to reduce the computational time. The DIZK utilizes Apache Spark to parallelize certain computations in zk-SNARK. We can distribute computation tasks evenly across multiple DSNs, enabling efficient distributed computing. Therefore, by decomposing the computations with DIZK, we can reduce the overall time required for proof generation.

C. Performance Analysis of Double Compression

1) Computational Overheads: Fig. 7 shows the computational overheads of EDCOMA with regard to two double compression ratios, namely D-compression ratio r_D and P-compression ratio r_P. The file size in the experiments is 100 MB. As shown in Fig. 7(a), as both the P-compression ratio r_P and D-compression ratio r_D increase, the computation time of the Store algorithm will first rapidly and then slowly decrease. The reason is that the increase of either r_D or r_P results in the compression of files and the decrease of number of data authenticators, which thereby greatly reduces the computational time in the Store algorithm.

Fig. 7(b) illustrates the computation time of the Prove algorithm. We can see that as the P-compression ratio r_P increases, the computation time of the Prove algorithm remains unchanged. On the other hand, as the D-compression ratio r_D increases, the computation time greatly decreases. This is because the computational cost of the Prove algorithm majorly lies in the calculation of the proof of the compression arithmetic circuit in ZKP, while the computation cost of generating other auditing proofs is much lower compared to circuit proof generation. As P-compression ratio r_P increases, since the total challenged file size (the number of all challenged data blocks) and r_D do not change, the computational time of circuit calculation as well as the overall computation time remains unchanged. As D-compression ratio r_D increases, the number of commitments in the compression arithmetic circuit decreases, resulting in a decrease in computation time of the Prove algorithm.

Fig. 7(c) shows the computational overhead of the Verfiy algorithm in EDCOMA. As P-compression ratio r_P increases, when D-compression ratio r_D is relatively small (e.g., 20 or 30), the computational cost shows a trend of first decreasing and then increasing. When D-compression ratio r_D exceeds 60, the computational cost will show a continuously upward trend. The reason of this phenomenon is that when D-compression ratio r_D is relatively small, both the number of data blocks per file chunk (determined by P-compression ratio r_P) and the number of file chunks have significant impact on the computation complexity of the Verify algorithm. In the first stage, when the number of data blocks is less than the number of file chunks, the verification time will decrease as the number of file chunks decreases. In the second stage, when the number of data blocks exceeds the number of file chunks, the processing time of data blocks dominates the verification time. Therefore, as the number of data blocks increases, the verification time starts to increase after a certain number of blocks. As D-compression ratio r_D increases to more than 60, the decrease in the first stage will become less apparent. As a result, the verification time directly increases whatever the number of data blocks is.

To sum up, we can see that as D-compression ratio r_D increases, the computational time of all three main algorithms decreases. However, performance trends become different in algorithms as P-compression ratio r_P increases. This is majorly because different components of computation operations (e.g., polynomial calculation, circuit calculation, exponentiation and multiplications in group) in the Store, Prove and Verify algorithms. Different components make these algorithms be impacted by P-compression ratio in various ways, which are discussed aforementioned in each sub-figure in detail.

2) Storage Overheads: To demonstrate the storage efficiency of EDCOMA brought by the double compression, we evaluate the storage overheads by comparing the storage cost of EDCOMA with [18] and [16], and analyzing the storage costs versus two compression ratios. Fig. 8(a) shows the comparison of additional storage costs of our proposed EDCOMA ($r_D = 10$ and $r_D = 20$), [18], and [16] against two compression ratios. From the figure, it can be seen that compared to other schemes, EDCOMA incurs the lowest storage overheads. When P-compression ratio r_P is 10, the storage overheads of EDCOMA with $r_D = 10$ and $r_D = 20$ are approximately 6 M and 3 M, respectively. In contrast, the storage costs of [18] and [16] exceed 60 M. This is because we use double compression method to significantly reduce the number of data authenticators. Furthermore, as the P-compression ratio of EDCOMA increases, the storage costs further decrease, which can be seen more clearly in Fig. 8(b).

Fig. 8(b) illustrates the storage overheads of EDCOMA versus both D-compression ratio r_D and P-compression ratio r_P. It can be observed that as D-compression ratio r_D or P-compression ratio r_P increases, the size of data authenticators in EDCOMA

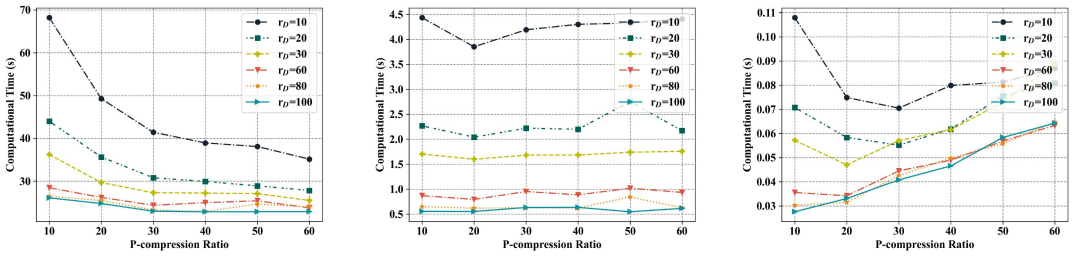


Fig. 7. Performance of EDCOMA in terms of different compression ratios. (a) Computational time of the Store algorithm. (b) Computational time of the Prove algorithm. (c) Computational time of the Verify algorithm.

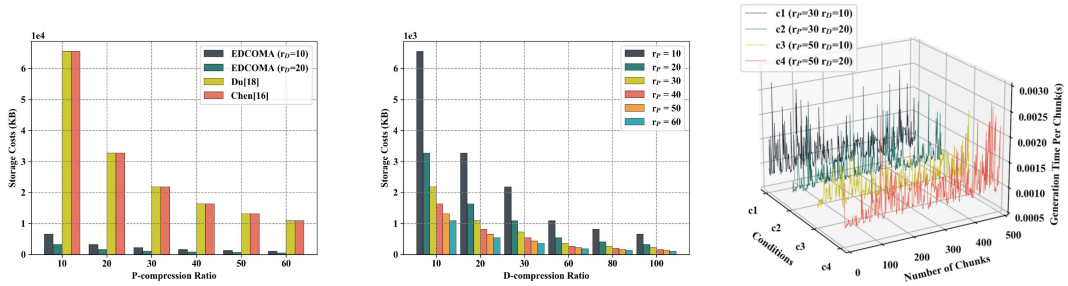


Fig. 8. (a) Comparison of storage cost of [16], [18] and EDCOMA. (b) Storage cost of EDCOMA versus different compression ratios. (c) Computational time per data authenticator in the Store algorithm against different number of chunks and compression ratio conditions.

greatly decreases. This is because the increase of the two ratios of double compression can both results in the compression and decrease of the overall size of data authenticators, which thereby saves the storage overheads.

3) Authenticator Generation: As demonstrated in Fig. 8(c), we test the computation time of each authenticator generation versus different chunk number and D-compression/P-compression ratios. We find that the generation time is fluctuant and mostly falls within the range of 0.0005 s to 0.003 s under different conditions. Generally, the computational time is limited within a certain range and won't last too long in the calculation of any single file chunk. The usage of double compression can contribute to the stable generation of the data authenticator since more data blocks are compressed and summarized into one authenticator.

4) Auditing Probability: EDCOMA adopts random challenge strategy that randomly selects file chunks to be audited. Fig. 9(a) demonstrates the auditing probability of EDCOMA versus different number of challenged file chunks, P-compression ratio, and D-compression ratio.

In Fig. 9(a), the auditing probability is affected by both the number of challenged chunks and two compression ratios. The corruption ratio ε is set as 0.1%. We can see that the proposed EDCOMA achieves a high auditing probability that approaches 100% as the P-compression ratio r_P and the number of challenged file chunks increase. The auditing probability will approach faster to 100% with the increase of D-compression ratio r_D, which indicates that the scheme with a higher D-compression ratio can achieve over 99.9% probability earlier than the scheme with a lower D-compression ratio. In addition, the experiment results demonstrate that in order to attain a high auditing probability, we only need to challenge a limited number of file chunks. Conversely, it is also feasible to enhance the auditing probability by adjusting the parameters.

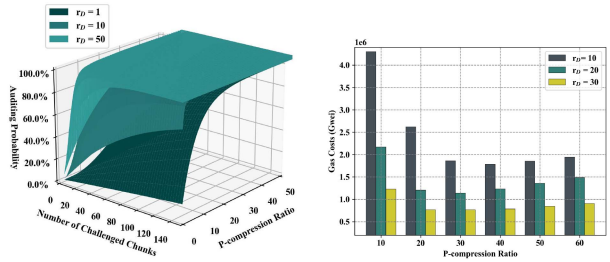


Fig. 9. (a) Auditing probability of EDCOMA with corruption ratio $\varepsilon = 0.1\%$. (b) Gas cost of EDCOMA versus different compression ratios.

D. On-Chain Gas Cost

We construct a private test network based on Ethereum, and deploy our smart contract implemented in Solidity language of version 0.8.7. We test the gas costs of verification in SC and calculate the ether costs by $Cost_{ETH} = Gas \times Price_{Gas}$. The gas price is set at 0.001 ether per million gas. We compute the corresponding USD cost based on the ether price of $1200.52 in January 2023.

The gas cost of the storage contract (SC) in EDCOMA on-chain auditing is presented in Fig. 9(b). As the D-compression ratio r_D increases, the gas cost of SC decreases. This is because in the case of fixed auditing probability, the higher the D-compression ratio r_D is, the fewer data authenticators and file blocks need to be checked, which leads to the lower gas cost of verification in SC. However, with the increase of P-compression ratio r_P, the gas cost of SC shows a trend of first decreasing and then increasing. This is because as the P-compression ratio r_P increases, the number of challenged file chunks decreases. This reduces the gas cost of the authenticator verification, which is the major cost when the P-compression ratio r_P stays relatively low within a certain range (e.g., 10–30). However, when the P-compression ratio r_P exceeds this range (e.g., higher than 30), the gas cost of generating Φ_P becomes dominant and results in the increase of the overall cost. From Fig. 9(b), we can also see that the highest gas cost consumption verified by this scheme is about 4.3×10^6 (5.1 dollars), and the lowest gas cost consumption is 7.6×10^5 (0.9 dollars) with the auditing probability over 99.9%, indicating that EDCOMA can audit DS in high probability with low costs.

Fig. 9(b) illustrates the gas cost for different compression ratios under a file size of 10 MB. It is worth noting that this file size refers to challenged set instead of the stored files. Due to the auditing probability demonstrated in Theorem 4 and Fig. 9(a), the challenge set for verification can maintain small, while the stored files can be much larger. Furthermore, the gas cost, which ranges from 0.9 to 5 dollars, can be adjusted by varying the compression ratios. Even at the lowest cost of 0.9 dollars, a 99.9% auditing probability can be achieved, thereby meeting requirements of most practical scenarios.

VII. Conclusion

Guaranteeing the data integrity in decentralized storage nodes (DSNs) is crucial to a DS platform and is a major challenge in DS. In this work, we propose an efficient double compressed auditing scheme EDCOMA for blockchain-based decentralized storage, which can achieve both storage efficiency and lightweight storage preparation. Thanks to the design of a double compression method, we build a light-weight data authenticator and minimize the extra storage cost of DSNs. Due to the double compression method, the performance of pre-processing and verification can also be greatly improved. Moreover, we design a compression arithmetic circuit using ZKP to verify the execution of compression operations in DSNs and prevent the possible replay attack from the nodes. Security analysis proves that EDCOMA is secure under the random oracle model. We implement a prototype of EDCOMA and conduct extensive experiments, which demonstrates the superiority of EDCOMA to the state-of-the-art in both computation and storage. Experimental results show that compared with the state-of-the-art, EDCOMA can reduce 92.9% computation cost in auditing preparation and 89.9% additional storage cost on average.

In future work, we will focus on improving ZKP protocol by reducing the computational time of proof generation. The proof generation algorithm in EDCOMA is currently executed solely on a single DSN, failing to fully exploit the advantages of the decentralized environment. Future approaches will consider parallelizing computation among DSNs to reduce the cost. We will explore the feasibility of parallelizing the circuit as well as ZKP protocol across multiple DSNs to further improve their performance in proof generation.

References

[1] N. Satoshi, "Bitcoin a peer-to-peer electronic cash system," White Paper, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[3] N. Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *J. Netw. Comput. Appl.*, vol. 162, 2020, Art. no. 102656.

[4] J. Benet and N. Greco, "Filecoin: A decentralized storage network," *Protoc. Labs*, Original Filecoin White Paper, Jul. 2018.

[5] D. Vorick and L. Champine, "Sia: Simple decentralized storage," Nebulous Inc, White Paper, Nov. 2014.

[6] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," Storj Inc, White Paper, Dec. 2014.

[7] E. Bacis, S. D. C. di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati, "Securing resources in decentralized cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 286–298, 2020.

[8] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 824–835, Sep./Oct. 2019.

[9] K. Fan, Z. Bao, M. Liu, A. V. Vasilakos, and W. Shi, "Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT," *Future Gener. Comput. Syst.*, vol. 110, pp. 665–674, 2020.

[10] N. Garg, S. Bawa, and N. Kumar, "An efficient data integrity auditing protocol for cloud computing," *Future Gener. Comput. Syst.*, vol. 109, pp. 306–316, 2020.

[11] H. Yu et al., "Efficient dynamic multi-replica auditing for the cloud with geographic location," *Future Gener. Comput. Syst.*, vol. 125, pp. 285–298, 2021.

[12] B. Sengupta and S. Ruj, "Efficient proofs of retrievability with public verifiability for dynamic cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 138–151, First Quarter 2020.

[13] H. Yu, Z. Yang, S. Tu, M. Waqas, and H. Liu, "Blockchain-based offline auditing for the cloud in vehicular networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 2944–2956, Sep. 2022.

[14] L. Zhou, A. Fu, Y. Mu, H. Wang, S. Yu, and Y. Sun, "Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system," *Inf. Sci.*, vol. 545, pp. 254–276, 2021.

[15] T. Wu, G. Yang, Y. Mu, F. Guo, and R. H. Deng, "Privacy-preserving proof of storage for the pay-as-you-go business model," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 563–575, Mar./Apr. 2021.

[16] R. Chen, Y. Li, Y. Yu, H. Li, X. Chen, and W. Susilo, "Blockchain-based dynamic provable data possession for smart cities," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4143–4154, May 2020.

[17] H. Duan, Y. Du, L. Zheng, C. Wang, M. H. Au, and Q. Wang, "Towards practical auditing of dynamic data in decentralized storage," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 708–723, Jan./Feb. 2023.

[18] 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. 3038–3054, Sep./Oct. 2022.

[19] N. Koblitz and A. J. Menezes, "The random oracle model: A twenty-year retrospective," *Des. Codes Cryptogr.*, vol. 77, no. 2, pp. 587–610, 2015.

[20] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *J. ACM*, vol. 51, no. 4, pp. 557–594, 2004.

[21] J. Benet, "IPFS-content addressed, versioned, P2P file system," Original IPFS White Paper, Jul. 2014, doi: 10.48550/arXiv.1407.3561.

[22] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, Springer, 1988, pp. 369–378.

[23] G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[24] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[25] 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.

[26] Y. Su, Y. Li, B. Yang, and Y. Ding, "Decentralized self-auditing scheme with errors localization for multi-cloud storage," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2838–2850, Jul./Aug. 2022.

[27] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, "Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1118–1132, Mar./Apr. 2022.

[28] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[29] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Serv. Comput.*, vol. 14, no. 1, pp. 71–81, Jan./Feb. 2021.

[30] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Inf. Sci.*, vol. 519, pp. 348–362, 2020.

[31] J. Li, J. Wu, G. Jiang, and T. Srikanthan, "Blockchain-based public auditing for Big Data in cloud storage," *Inf. Process. Manage.*, vol. 57, no. 6, 2020, Art. no. 102382.

[32] C. Wang, S. Chen, Z. Feng, Y. Jiang, and X. Xue, "Blockchain-based data audit and access control mechanism in service collaboration," in *Proc. IEEE Int. Conf. Web Serv.*, 2019, pp. 214–218.

[33] Y. Miao, Q. Huang, M. Xiao, and W. Susilo, "Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 3663–3676, 2022.

[34] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in P2P cloud storage," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 561–568.

[35] G. Ateniese, L. Chen, M. Etemad, and Q. Tang, "Proof of storage-time: Efficiently checking continuous data availability," in *Proc. 27th Annu. Netw. Distrib. Syst. Secur. Symp.*, The Internet Society, 2020.

[36] H. Yu, Q. Hu, Z. Yang, and H. Liu, "Efficient continuous big data integrity checking for decentralized storage," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1658–1673, 2021.

[37] Y. Du, H. Duan, A. Zhou, C. Wang, M. H. Au, and Q. Wang, "Towards privacy-assured and lightweight on-chain auditing of decentralized storage," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 201–211.

[38] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2008, pp. 90–107.

[39] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," *Commun. ACM*, vol. 59, no. 2, pp. 103–112, 2016.

[40] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. 35th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Vienna, Austria, Springer, 2016, pp. 305–326.

[41] A. K. Kasgar, J. Agrawal, and S. Shahu, "New modified 256-bit MD5 algorithm with SHA compression function," *Int. J. Comput. Appl.*, vol. 42, no. 12, pp. 47–51, 2012.

[42] H. Miyaji, A. Kawachi, and A. Miyaji, "String commitment scheme with low output locality," in *Proc. 14th Asia Joint Conf. Inf. Secur.*, 2019, pp. 32–39.

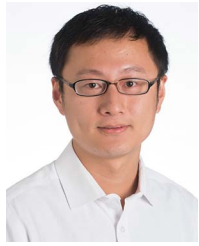
[43] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Proc. 16th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Singapore, Springer, 2010, pp. 177–194.

[44] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptographic protocols: A survey," *Cryptol. ePrint Arch.*, Cryptology Research Group, Rep. no. 2004/064, Jun. 2004. [Online]. Available: https://eprint.iacr.org/2004/064

[45] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Proc. 11th Int. Workshop Fast Softw. Encryption*, Delhi, India, Springer, 2004, pp. 371–388.

[46] G. Botrel, T. Piellard, Y. El Housni, I. Kubjas, and A. Tabaie, "Consensys/gnark: V0. 7.0," *ConsenSys Ventures*, Report, Mar. 2022, doi: 10.5281/zenodo.5819104.

[47] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "DIZK: A distributed zero knowledge proof system," in *Proc. 27th USENIX Secur. Symp.*, Baltimore, 2018, pp. 675–692.

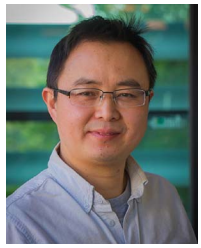


Haiyang Yu (Member, IEEE) received the PhD degree in computer science and technology from the Beijing University of Technology, in 2019. From 2017 to 2018, he visited the University of Melbourne in Australia. He has authored or coauthored more than 20 papers in highly ranked journals and top conference proceedings. He is currently an associate professor of computer science with the Beijing University of Technology. His research interests include cloud storage auditing, blockchain, and federated learning. He is a member of ACM.

Yurun Chen (Student Member, IEEE) received the BS degree in computer science and technology from Jiangsu University, Jiangsu, China, in 2022, and the BS degree in computer science from California State University San Bernardino, in 2022. He is currently working toward the master's degree of computer technology with the Beijing University of Technology, Beijing, China. From 2019 to 2021, he studied with California State University San Bernardino. His research interests include storage auditing and blockchain.

Zhen Yang (Member, IEEE) received the PhD degree in signal processing from the Beijing University of Posts and Telecommunications, Beijing, China. He is currently the full professor of computer science and engineering with the Beijing University of Technology, Beijing, China. He has authored or coauthored more than 30 papers in highly ranked journals and top conference proceedings. His research interests include data mining, machine learning, trusted computing, and content security. He is the senior member of the Chinese Institute of Electronics.

Yuwen Chen (Member, IEEE) received the MS degree in computer software and theory from Zhengzhou University, Zhengzhou, China, in 2015, and the PhD degree in telematic engineering form the Technical University of Madrid, Madrid, Spain, in 2019. He is a lecturer with the School of Computer Science, Beijing University of Technology, Beijing, China. His research interests include IoT security and privacy, collaborative learning, model inversion, and gradient inversion.



Shui Yu (Fellow, IEEE) received the PhD degree from Deakin University, Melbourne, VIC, Australia, in 2004. He is currently a professor with the School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia. He has authored or coauthored four monographs and edited two books, and more than 400 technical papers, including top journals and top conferences, such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE/ACM Transactions on Networking*, and INFOCOM. His research interests include Big Data, security and privacy, and networking. He initiated the research field of networking for Big Data, in 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. He was an associate editor for the *IEEE Transactions on Parallel and Distributed Systems*. He is currently on a number of prestigious editorial boards, including *IEEE Communications Surveys and Tutorials* (area editor), *IEEE Communications Magazine*, and *IEEE Internet of Things Journal*. He was a distinguished lecturer of IEEE Communications Society during 2018–2021. He is a distinguished visitor of IEEE Computer Society, a voting member of IEEE ComSoc Educational Services board, and an elected Member of the Board of Governor of IEEE Vehicular Technology Society.