A Blockchain-Based Privacy-Preserving Scheme for Integrity Verification and Fair Payment in Cloud Data Storage

Li Zhenxiang[1], Jin Yuanrong[2], Mohammad Nazir Ahmad[3]
Sichuan Vocational College of Information Technology, Guangyuan, China[1, 2]
Infrastructure University Kuala Lumpur, Kuala Lumpur, Malaysia[1, 2]
Institute of Visual Informatics, Universiti Kebangsaan Malaysia, Malaysia[3]

Abstract—**Ensuring the integrity of outsourced data in cloud storage remains a critical challenge, especially when existing auditing schemes rely on centralized third-party auditors (TPAs), which introduce single points of failure, privacy leakage risks, and a lack of economic fairness. Current blockchain-based approaches improve transparency but still fail to simultaneously achieve privacy-preserving verification and fair payment between data owners and cloud service providers (CSPs). To address this gap, this study proposes a blockchain-based integrity verification scheme that supports decentralized, privacy-preserving, and economically fair audits for encrypted cloud data. The proposed scheme integrates homomorphic linear authenticators (HLA) and multi-party computation (MPC) to verify data integrity without revealing plaintext, while smart contracts are used to enforce automatic payment or penalty based on audit results, ensuring fairness and accountability. A prototype implementation confirms the practicality of the system. Experimental results show that the audit latency is reduced by up to 35 per cent and smart contract gas consumption by approximately 30 per cent compared to existing schemes, while maintaining low computation and communication overhead. Security analysis demonstrates that the scheme provides data integrity, privacy protection, fairness, and resistance to replay and collusion attacks. Overall, this work offers a practical and scalable solution for secure cloud storage auditing.**

Keywords—Cloud storage; blockchain; integrity verification; smart contract; privacy-preserving audit; fair payment

I. Introduction

Cloud storage has become an essential component of modern data management infrastructures [1]. However, outsourcing data to third-party cloud service providers (CSPs) raises critical concerns regarding data integrity, confidentiality, and user trust [2]. Since users no longer possess physical control over their data, they require reliable mechanisms to ensure that outsourced files remain intact and unmodified. Traditional auditing schemes rely on trusted third-party auditors (TPAs), but these centralized architectures suffer from inherent weaknesses such as single points of failure, implicit trust assumptions, potential collusion, and limited transparency [3].

Blockchain technology provides a decentralized and tamper-resistant alternative for cloud auditing, replacing TPAs with distributed consensus and verifiable smart contract execution [4], [5]. Although several blockchain-based integrity verification schemes have been proposed, most of them still exhibit one or more of the following limitations:

- They focus on transparency but fail to protect data privacy during audits.
- They do not support fair economic compensation between users and CSPs in cases of data loss or fraud.
- They provide incomplete support for dynamic data operations or require additional trust in off-chain verifiers [6].

Therefore, an important research gap remains: existing approaches do not offer a unified framework that simultaneously achieves decentralized verification, privacy preservation, and fair payment in cloud storage environments.

To address this gap, this study proposes a blockchain-based integrity verification scheme that integrates privacy-preserving audit and economic fairness into a single framework. Homomorphic Linear Authenticators (HLA) and Multi-Party Computation (MPC) are combined to enable verification of encrypted data without revealing its content. In addition, smart contracts are employed to enforce an automatic fair payment mechanism that compensates CSPs only when data integrity is successfully proven, while also providing dispute resolution in case of corruption or data loss.

The main contributions of this work are summarized as follows:

- We design a fully decentralized cloud data integrity verification framework that removes reliance on TPAs and guarantees audit correctness while preserving data privacy.
- We develop a smart contract–based fair payment protocol that enforces automatic rewards or penalties based on audit outcomes, ensuring economic accountability between users and CSPs.

- We implement the prototype and conduct comprehensive security analysis and experimental evaluation. Results demonstrate that the proposed scheme achieves low audit latency, reduced communication overhead, and feasible gas consumption in practical cloud environments.

The rest of the study is organized as follows: Section II reviews related work. Section III introduces the system architecture and threat model. Section IV presents the proposed scheme in detail. Section V provides security analysis. Section VI discusses performance evaluation and experimental results. Finally, Section VII concludes the study and outlines future research directions.

II. Related Work

Ensuring the integrity of outsourced data has long been a fundamental challenge in cloud storage systems. Early cryptographic solutions such as Provable Data Possession (PDP) and Proof of Retrievability (PoR) enabled users or trusted third-party auditors (TPAs) to verify data correctness without downloading the entire file [7]. However, these methods rely heavily on centralized auditors, which introduces risks of single-point failure, collusion, and trust assumption issues [8].

To mitigate reliance on TPAs, blockchain-based integrity verification schemes have gained increasing attention. Zhang et al. combined blockchain with lattice-based cryptography to achieve public auditing and improved transparency, but their scheme lacks support for efficient computation and dynamic data updates, which limits real-world applicability [9]. Zhou et al. proposed a blockchain-enabled data monitoring scheme; however, it provides limited analysis of computational costs and does not address economic fairness between cloud service providers (CSPs) and users [10].

Recent works have explored the integration of homomorphic authenticators and advanced cryptographic techniques to enhance verifiability. Sankar et al. introduced a lattice-based multi-cloud auditing model, which offers theoretical security but lacks practical deployment evaluation [11]. Xie et al. proposed a T-Merkle hash tree to support dynamic data operations; nevertheless, their design does not incorporate fair payment or dispute resolution mechanisms [12]. Liu and Huang presented privacy-preserving and blockless verification schemes, but challenges remain in achieving decentralized enforcement and secure economic incentives [13], [14].

Another research direction focuses on blockchain-based economic mechanisms for cloud data services. While these methods record audit outcomes on-chain, most do not leverage smart contracts for automatic enforcement or compensation. For example, He et al. improved communication efficiency but failed to establish a clear arbitration mechanism for disputes [15]. Li et al. incorporated zero-knowledge proof-based privacy, yet did not support on-chain payment settlement or dynamic file updates [16].

Wang et al. developed a smart contract-driven integrity auditing scheme for multi-cloud and multi-replica environments, optimizing tag generation to reduce overhead [17]. However, their system assumes a trusted key generation center (KGC) and primarily focuses on multi-CSP coordination rather than privacy preservation and fair exchange in a single-cloud scenario.

In summary, existing blockchain-based auditing schemes still face the following limitations:

- Reliance on centralized authorities such as TPAs or KGCs.
- Absence of fair payment mechanisms to guarantee accountability between users and CSPs.
- Incomplete support for encrypted data verification while preserving privacy.

To overcome these challenges, this study proposes a decentralized, privacy-preserving integrity verification scheme that integrates homomorphic authenticators, multi-party computation (MPC), and smart contracts to achieve secure auditing, automated dispute resolution, and fair compensation.

III. Scheme Framework

A. Scheme Model

The proposed scheme consists of three primary entities, as illustrated in Fig. 1.

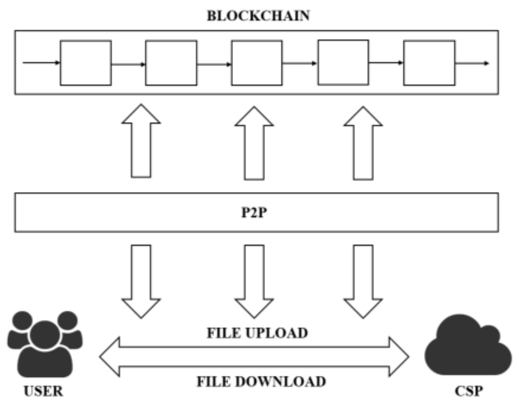

Fig. 1. Scheme model.

1) Data owner (DO): A user who outsources encrypted data to the cloud and later requests integrity verification. The DO is responsible for generating encryption keys, computing homomorphic tags, and initiating audit requests.

2) Cloud service provider (CSP): A storage service provider that maintains outsourced data on behalf of the DO. The CSP responds to audit challenges by generating proofs of data possession.

3) Blockchain network: A decentralized ledger that hosts smart contracts. It facilitates audit challenge distribution, proof aggregation, verification, and fair payment based on audit results. The blockchain is assumed to be transparent, tamper-resistant, and always available.

The system is designed to operate without relying on any third-party auditor or centralized key generation authority. All integrity audits, payment mechanisms, and verification processes are executed through smart contracts deployed on the blockchain. Data remains encrypted throughout the process using AES-GCM, while integrity verification is performed using homomorphic linear authenticators (HLA).

B. Threat Model

We consider both internal and external adversaries.

- Malicious CSP: The primary threat arises from a dishonest CSP, who may:
  - Delete or alter outsourced data to reduce storage overhead.
  - Forge integrity proofs to pass audits without storing actual data.
  - Reject fair payment conditions or evade penalties following a failed audit.
- External Attacker: An outsider may attempt to:
  - Intercept or tamper with communication between DO and CSP.
  - Replay previous audit responses to bypass verification.
  - Eavesdrop on blockchain transactions to infer private information.
- Collusion Risk: The CSP may attempt to collude with external entities to manipulate audit results or payment transactions. However, smart contracts on the blockchain act as impartial executors, preventing such collusion.

We assume that the blockchain platform is secure, the consensus mechanism is reliable, and all cryptographic primitives (e.g., AES-GCM, Keccak-256, HLA) are resilient against known attacks. All communication channels are assumed to be protected by standard TLS protocols.

C. Security Targets

The proposed scheme aims to achieve the following security objectives:

1) Data integrity: CSPs should be unable to modify or delete data without detection. Audit verification must fail if the data is tampered with.

2) Privacy preservation: The CSP cannot learn the content of the data during upload, verification, or audit, as all data remains encrypted and audit is performed via MPC and HLA.

3) Audit correctness: If the CSP stores data correctly and follows the audit protocol honestly, the verification will succeed with high probability.

4) Fair payment: The smart contract ensures that CSPs are compensated only when audits are successful and penalized when they fail to produce valid proofs.

5) Collusion resistance: Neither the CSP nor any external party can manipulate or falsify audit results due to the tamper-proof nature of the blockchain and the transparency of smart contract execution.

6) Replay resistance: Each audit challenge is randomly generated and time-bound, preventing the reuse of old responses.

IV. Scheme Realization

This section outlines the proposed scheme's implementation process, which is divided into five main phases. It begins by introducing the notations and initialization settings that establish the theoretical foundation of the scheme. Following this foundational setup, the focus shifts to a detailed examination of the three critical parts: file upload, file download, and file update.

A. Notations

The symbols used throughout the scheme are defined in Table I.

TABLE I. NOTATIONS

Notation	***Definition***
F	File
C	Ciphertext
B_i	i-th data block
S_j	j-th share in MPC
P_j	j-th party in MPC
M	MPC protocol
R	Merkle Hash Tree Root Node
n	Number of data blocks
ID	Identifier
σ	Homomorphic tag
Pay_{stor}	Storage service fees
Pay_{comp}	Compensation costs

B. Initial Setting

The proposed scheme leverages several core cryptographic components and concepts:

1) Cyclic groups and bilinear mapping: Let G and G_T be two cyclic groups of the same prime order p, with g as the generator of group G. A bilinear mapping is defined as, $e: G \times G \rightarrow G_T$. These form the foundation for constructing the cryptographic scheme.

2) User and file identity: Each user U is identified by an ID_U and a corresponding Ethereum account EA_U. Files are identified by an ID_F. The Cloud Service Provider (CSP) is also identified by an ID_{CSP} and has a corresponding Ethereum Account EA_{CSP}.

3) Encryption keys and signatures: User U randomly generates an RSA key pair (spk, ssk) for digital signature purposes. Additionally, user U selects an element u from G, with a private key $ssk = x \in Z_P$, and calculates the public key $y = g^x$.

4) Multi-Party Computation (MPC): To enhance the security and trustworthiness of the integrity verification process, MPC is employed. In MPC, the data is divided into multiple shares S_j, which are distributed among different parties P_j. These parties collaboratively perform computations under the MPC protocol without disclosing their individual inputs.

5) Economic model: Upon successful storage of user data, users are required to pay the CSP storage fees Pay_{stor}. If user data is lost or damaged, the CSP must provide compensation to the user, denoted as Pay_{comp}.

These cryptographic elements and concepts provide the necessary theoretical and technical foundation for the proposed blockchain-based cloud data integrity verification scheme, ensuring data security and transaction fairness.

C. File Upload

There are a total of six steps for file upload, as shown in Fig. 2.

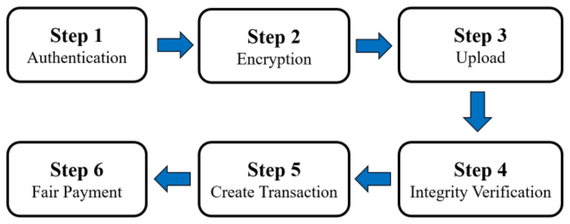

Fig. 2. Steps of file upload.

1) Step-1: Authentication: Before uploading file F, the CSP must verify the user's identity. The user U uses a digital certificate issued by a trusted Certificate Authority (CA) to verify their identity. Specifically, user U first hashes their identity as $h\ (ID_U)$ and then signs this hash using their private key to create the signature $Sig_{ssk}\ (h\ (ID_U))$. The user U sends this signature along with their digital certificate to the CSP. Upon receiving the signature and the certificate, the CSP uses the CA's public key to validate the digital certificate, ensuring its authenticity. Then, CSP uses the public key spk from the certificate to verify the signature . If the verification is successful, the user is authenticated and can proceed to the step-2.

2) Step-2: Encryption: User U generates a signature for file F using the private key ssk.

$$Sig_F = ID_F \parallel Sig_{ssk}(ID_F) \quad (1)$$

Sig_F can be used as an identifier for file F. Subsequently, user U divides the file F into n data blocks, $F = \{B_i\}(1 \le i \le n)$. For the i-th data block B_i, user U uses PBKDF2 algorithm to calculate the convergence key and uses AES-GCM algorithm to calculate the ciphertext.

Generate a unique salt value P_i for each data block B_i.

$$K_{B_i} = PBKDF2(B_i, P_i, iterations, dklen) \quad (2)$$

Encrypt the data block using the derived key K_{B_i}.

$$C_{B_i} = AES\text{–}GCM(K_{B_i}, B_i) \quad (3)$$

User U calculates file tags, using Keccak-256 hash function.

$$T_F = Keccak - 256(C_{B_i}) \quad (4)$$

User U uses data block ciphertext $\{C_{B_i}\}$ to construct a Merkle hash tree and calculate its root points.

$$R_C = MerkleTree(\{C_{B_i}\}) \quad (5)$$

User U calculates file signature.

$$\delta = H(R_C)^{x \bmod p} \quad (6)$$

User U calculates homomorphic signature.

$$W_i = ID_F \parallel i \quad (7)$$

$$\sigma_i = (H(W_i \times u^{C_{B_i}}))^{x \bmod q} \quad (8)$$

3) Step-3: Upload: User U sends the file label T_F to CSP, and CSP performs repeatability detection. If the file label T_F already exists on the cloud server, it indicates duplication. In this case, CSP and the user execute a proof of ownership agreement to prove that the user indeed owns the file F, and the user does not need to upload the file F again. If the file label T_F is not stored on the cloud server, it indicates that the file F is not duplicated, and user U will send $(Sig_F, \{C_{B_i}\}, \{\sigma_i\})$ to CSP.

4) Step-4: Integrity verification: To enhance the security and privacy of the integrity verification process, user U can act as an auditor and engage in an integrity verification protocol with the Cloud Service Provider (CSP), leveraging Multi-Party Computation (MPC) as follows:

User U retrieves the file identifier $Sig_F = ID_F \parallel Sig_{ssk}(ID_F)$ from the cloud server.

Using the public key spk to verify the correctness of the signature $Sig_{ssk}(ID_F)$.

If the verification fails, the user will output a result of false. If the verification is successful, the user can recover the file identification ID_F.

Initialization: User U selects a subset $I = \{S_1, S_2, S_3, \ldots, S_{c-1}, S_c\} \subset [1, n]$ containing c elements. This subset is used as the seed value set $\{S_i\}$ to initiate M.

Sharing Data for MPC: User U divides the data into shares S_j for the selected indices I. These shares are distributed among different parties P_j involved in M.

Engaging the MPC Protocol: The parties $\{P_j\}$ collaboratively execute M to securely compute random numbers. The protocol ensures that none of the parties, including the CSP, can learn the individual shares or the final random numbers until the computation is complete.

Secure Computation: M employs a pseudo-random function, denoted as $f: \{0,1\}^* \to [1, n]$, to securely compute the random numbers. The function takes as input the shares $\{S_j\}$, the current block identifier B_{id} and the current timestamp T_s.

Output Generation: As a result of the secure computation, M outputs two sets of random numbers: $\{r_i\}$ and $\{z_i\}$. These numbers are generated in such a manner that they are unpredictable to outside observers, ensuring the integrity of the computation process.

Challenge Formation: User U forms a challenge $chal = \{(r_i, z_i)\}$ using the random numbers generated by M. This challenge is then sent to the CSP to verify the integrity of cloud data.

CSP calculates the proof information μ and L using the received challenge, where:

$$\mu = \sum_{i \in I} z_i C_{B_i} \quad (9)$$

$$L = \prod_{i \in I} \sigma_{z_i}^{r_i} \quad (10)$$

CSP sends (μ, L) as proof to user U.

User U checks the correctness of the following equation to check the integrity of cloud data:

$$\Gamma_1 : e(L, g) = e((\prod_{i \in I} H(W_i)^{z_i} \cdot u^{\mu}), y) \quad (11)$$

This step utilizes MPC to ensure that the computation of μ and L is done without revealing the actual values of $\{C_{B_i}\}$ and $\{\sigma_{Z_i}\}$ to any single party, enhancing the privacy of the data.

By incorporating MPC into the integrity verification process, this approach not only maintains the confidentiality of the data during the verification but also enhances the security and trustworthiness of the cloud storage system. The use of MPC allows multiple parties to compute the verification without revealing their inputs, thereby providing a more secure and privacy-preserving verification process. Fig. 3 illustrates the process of step-4.

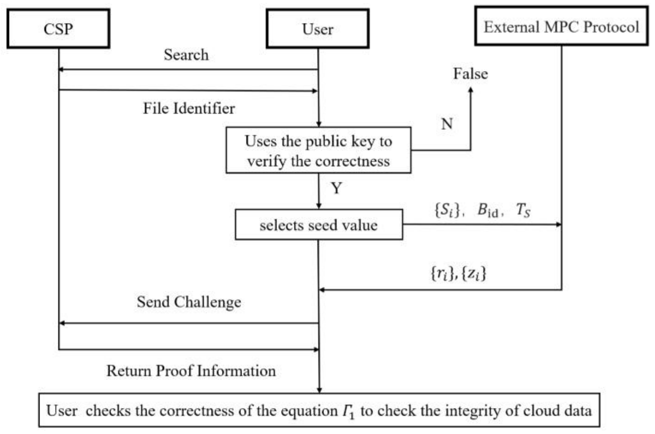
CSP
User
External MPC Protocol
Search
File Identifier
False
N
Uses the public key to verify the correctness
Y
$\{S_i\}$, B_{id}, T_S
selects seed value
$\{r_i\}, \{z_i\}$
Send Challenge
Return Proof Information
User checks the correctness of the equation Γ_1 to check the integrity of cloud data

Fig. 3. Step-4 integrity verification.

5) Step-5: Create transaction: User U can generate effective file tamper proof records by integrating auxiliary information from file F into a transaction in the blockchain. This allows other users to have a clear understanding of the storage service quality provided by CSP and helps them choose the most suitable storage service. The details are as follows. User U calculate:

$$Data = H(ID_U) \parallel H(ID_F) \parallel flag \cdot Pay \parallel \delta \parallel Aug_F \quad (12)$$

where, $flag \cdot Pay > 0$ indicates that user U pays storage service fees to CSP, $flag \cdot Pay < 0$ indicates that CSP pays compensation fees to user U, and Aug_F represents additional information for file F.

As shown in Fig. 4, the user submits Smart Contract 1 to generate a transaction TX, where the transaction TX parameter is $From = EA_U$,$T_o = EA_{CSP}$ and $Data = H(ID_U) \parallel H(ID_F) \parallel flag \cdot Pay \parallel \delta \parallel Aug_F$.

Smart Contract 1

```
Input:  $ID_U$; $ID_{CSP}$; $Data$
Output: TX
1: Compute $EA_U$;
2: Compute $EA_{CSP}$;
3: $From = EA_U$;
4: $T_o = EA_{CSP}$;
5: $Value = Gas$;
6: TX= $From||T_o||Value||Data$;
```

Fig. 4. Smart contract 1.

After receiving the output result of Smart Contract 1, User U uses the private key ssk to generate signature information $Signature = Sig_{ssk}(\text{TX})$, and submits the transaction $\text{TX} = From \parallel T_o \parallel Value \parallel Data \parallel Signature$ to the blockchain.

6) Step-6: Fair payment: After data is uploaded to the cloud server, users need to pay for the storage services provided by CSP. As shown in Fig. 5, user U can submit Smart Contract 2 to the blockchain, where Smart Contract 2 will be automatically activated and executed. If the formula Γ_1 is established, it means that the data block $\{B_{ij}\}$ has been fully stored on the cloud server, and at this time, user U needs to pay the corresponding storage service fee to CSP. Otherwise, if the equation Γ_1 is not valid, it indicates that the data block $\{B_{ij}\}$ has been damaged or lost, and CSP needs to take corresponding responsibility and provide certain compensation for the damage or loss of the data. The mechanism of fair payment is shown in Fig. 6.

Smart Contract 2

```
Input:  $ID_U$; $ID_{CSP}$; $Pay_{stor}$; $Pay_{comp}$
Output: $flag$
1:          Compute $EA_U$;
2:          Compute $EA_{CSP}$;
3:      If  $(\Gamma_1)$  then
4:          $Pay = Pay_{stor}$;
5:          $flag = 1$;
6:      Else
7:          $Pay = -Pay_{stor}$;
8:          $flag = -1$;
9:      Return  $flag$
```

Fig. 5. Smart contract 2.

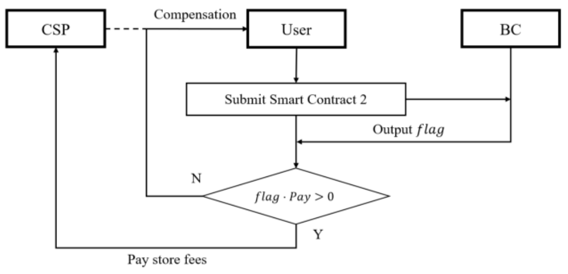

Fig. 6. Step-6 fair payment.

D. File Download

The user first submits a download request to the CSP and retrieves all encrypted data blocks. To verify the consistency of the data and restore the original file F, the process is as follows:

The encrypted data blocks are used as leaf nodes to construct a Merkle hash tree, from which the root node R_C is derived.

An external Multi-Party Computation (MPC) protocol is executed to verify the consistency of the data blocks. This step ensures a secure verification process without revealing any actual data to individual parties, while confirming that the blocks have not been tampered with.

The user then parses the transaction TX from the blockchain to extract the file identifier and associated verification metadata.

Verify the correctness of equation $\Gamma_2 : e(\delta, g) = e(H(R_C), y)$. If the equation Γ_2 holds true, it confirms the integrity and authenticity of the data blocks.

The decryption keys K_{B_i} are securely distributed among multiple parties using the MPC protocol. This ensures that no single party has access to the complete decryption key, enhancing the security of the decryption process.

With the securely distributed keys, decrypt each data block using the AES-GCM algorithm:

$$B_i = AES\text{–}GCM\text{–}decrypt(K_{B_i}, C_{B_i}) \tag{13}$$

Reassemble the original file $F = \{B_i\}$ from the decrypted data blocks B_i.

Fig. 7 shows the complete file download process.

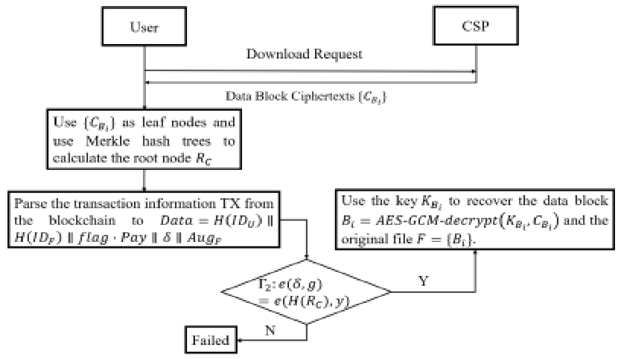
User
CSP
Download Request
Data Block Ciphertexts $\{C_{B_i}\}$
Use $\{C_{B_i}\}$ as leaf nodes and use Merkle hash trees to calculate the root node R_C
Parse the transaction information TX from the blockchain to $Data = H(ID_U) \parallel H(ID_F) \parallel flag \cdot Pay \parallel \delta \parallel Aug_F$
Use the key K_{B_i} to recover the data block $B_i = AES\text{-}GCM\text{-}decrypt(K_{B_i}, C_{B_i})$ and the original file $F = \{B_i\}$.
$\Gamma_2 : e(\delta, g) = e(H(R_C), y)$
Y
N
Failed

Fig. 7. File download.

E. File Update

When a data block B_i in file F needs to be updated to a new version, the process incorporates MPC to enhance privacy and security.

User U first encrypts the updated data block using the PBKDF2 key derivation function and the AES-GCM encryption algorithm to encrypt the updated data block B_i' to obtain the ciphertext C_i', and recalculates the homomorphic signature $\sigma_i' = (H(W_i) \cdot u^{C'_{B_i}}))^x$. Subsequently, user U will send an update request (i, C_i', σ_i') to CSP. CSP replaces the ciphertext block C_i in file F with the new block C_i' and send $\{\varpi_i\}$ to user U, where ϖ_i represents the information of the sibling nodes associated with the path from $H(C_i')$ to the root node.

User U first calculates the value R_C of the root node based on $\{\varpi_i\}$ and $H(C_i)$, and then continues to verify whether the condition $\Gamma_2 : e(\delta, g) = e(H(R_C)^x, g)$ holds. If it is true, it indicates that the position updated by the server is correct, that is, the ciphertext block C_i' is the updated block of ciphertext block C_i.

User U recalculates the value R_C' of the root node based on $\{\varpi_i\}$ and $H(B_i')$, and signs it $\delta' = H(R_C')^x$. Then, U recalculates the $Data = H(ID_U) \parallel H(ID_F) \parallel flag \cdot Pay \parallel \delta' \parallel Aug_F$, and finally integrates the $Data$ information into a blockchain transaction on the blockchain. The insertion and deletion operations of data blocks can refer to the data block update operations mentioned above.

V. Security Analysis

This section evaluates the proposed scheme against the adversarial threats defined in Section III and demonstrates how the system satisfies its key security objectives.

A. Data Integrity and Unforgetability

Theorem 1: If the user does indeed download the original uploaded data, then the user will be able to successfully pass data consistency verification, with the process further secured by MPC.

Proof: In this scheme, when users download encrypted data $\{CB_i\}$ from the CSP, it is necessary to verify the consistency of their data in order to recover the correct original plaintext F. To achieve this goal, users can utilize a Merkle hash tree to compute the root node R, where the root node R is determined by all data block ciphertexts $\{CB_i\}$ as leaf nodes. If any part of this information is altered, the value of R will also change accordingly. Users can verify the correctness of the following equation to detect data consistency.

If equation $\Gamma_2 : e(\delta, g)$ holds true, it indicates that the downloaded ciphertext information $\{CB_i\}$ corresponds to the originally uploaded data. Subsequently, the user can recover the original file through convergence keys. Otherwise, if equation Γ2 does not hold true, it signifies that during the downloading process, the ciphertext of data blocks $\{CB_i\}$ has been corrupted.

B. Privacy Preservation

Theorem 1: This scheme can ensure the privacy protection of user data, and adversary X cannot obtain any useful information about user data during the integrity verification process.

Proof: The parameters $\{r, z, \mu, \delta, \sigma\}$ involved in this scheme do not expose any information about the user-uploaded data blocks B_i. In particular, there is a security vulnerability in the integrity verification scheme based on HLA, where in the integrity verification process, adversary X would utilize the generation of aggregate signature $\mu' = \sum_{i \in I} z_i C_{B_i}$, by repeatedly selecting indices (i, z_i) to obtain specific linear combinations. Finally, by solving linear equations, adversary X could extract user information contained in the aggregate signature.

C. Fair Payment Guarantee

The smart contract logic enforces incentive compatibility between the DO and CSP. Both parties deposit tokens before an audit round. The contract releases:

- Payment to the CSP if the audit passes.
- Refund or penalty compensation to the DO if the audit fails.

This design ensures that CSPs cannot receive rewards without proving data possession, and DOs cannot cheat without valid evidence. All transactions and outcomes are verifiable and immutable on the blockchain ledger, providing accountability without external arbitration.

D. Resistance to Replay and Collusion

Each audit challenge is freshly generated via secure on-chain randomness (e.g., Keccak-based hash or block nonce). Since the challenge indices and weights are unpredictable, a CSP cannot reuse previous responses (replay resistance).

In addition, the smart contract acts as an impartial, decentralized entity. Any attempt by DO and CSP to collude would be visible on-chain. Moreover, data tags and Merkle roots are bound to each user and file instance, making it infeasible to reuse or share tags across users (collusion resistance).

E. Soundness and Completeness of Verification

The verification algorithm satisfies:

- Completeness: If the CSP honestly stores all data and follows the protocol, verification always passes.
- Soundness: If any challenged block is corrupted or missing, the aggregated tag will not satisfy the bilinear verification equation, and the audit will fail with overwhelming probability.

Under the random oracle model, the probability of passing an audit without the correct data is negligible.

VI. Evaluation and Results

In this section, we evaluate the performance of our proposed scheme in terms of computational overhead, communication cost, and scalability. We also compare it against existing schemes to demonstrate its efficiency and practicality.

A. Experimental Setup

All experiments were conducted on a machine with an Intel Core i7-11700 CPU, 16 GB RAM, running Ubuntu 22.04. The implementation was developed in Python and Solidity. The blockchain environment is a local Ethereum testnet using Ganache, and smart contracts were deployed using the Truffle scheme.

We use the following cryptographic primitives:

- AES-GCM (128-bit) for symmetric encryption.
- Keccak-256 for hashing and challenge randomness.
- HLA over elliptic curve groups (BLS12-381).
- PBKDF2 for key derivation from file ID.

We evaluate performance on files ranging from 10 MB to 1 GB, and measure:

- Tag generation time
- Audit-proof generation and verification time
- Encryption time
- MPC overhead (optional component)
- Gas cost of smart contract interactions

B. Tag Generation and Upload Performance

As shown in Fig. 8, the tag generation time grows linearly with the file size. For a 100 MB file, the average tag generation takes 1.2 seconds, which is significantly lower than Zhou et al. and Sankar, both of which exceed 2.5 seconds under similar settings [10][11].

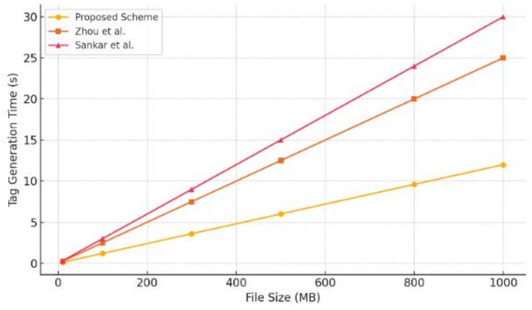

Fig. 8. Tag generation time vs. File size.

The AES-GCM encryption adds a fixed overhead of approximately 0.9 seconds per 100 MB, which remains acceptable in practice. Combined upload time remains under 2.5 seconds per 100 MB.

C. Audit Efficiency

Fig. 9 presents the time taken by the CSP to compute the audit proof and the time taken by the smart contract to verify it. Even for files of 500 MB, the audit process completes within 200 ms. The aggregated tag greatly reduces computational effort compared to verifying each block individually.

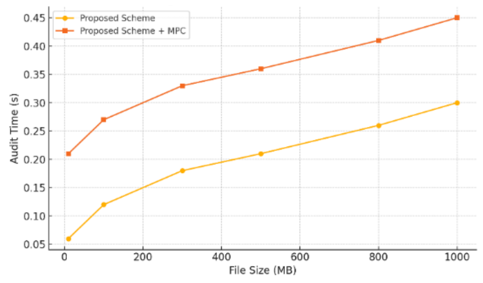

Fig. 9. Audit time vs. File size.

With MPC enabled, verification time increases modestly due to interaction rounds, reaching 420 ms for a 500 MB file. However, this added cost ensures privacy without revealing μ.

D. Data Update Overhead

Fig. 10 shows the cost of a single block update (modify/insert/delete). Tag recomputation and Merkle tree adjustment combined take less than 80 ms per block. On-chain update verification (including Merkle path) adds ~15,000 gas (~$0.03 equivalent on Ethereum).

This demonstrates that dynamic updates are practical and lightweight.

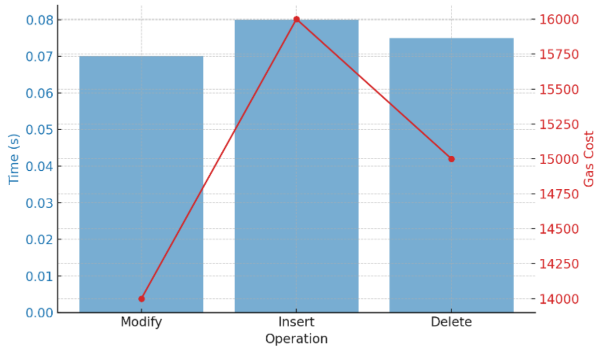

Fig. 10. Update overhead per operation.

E. Smart Contract Gas Cost

Fig. 11 shows the estimated gas consumption of core smart contract functions. Contract 1 handles audit-related operations, while Contract 2 governs fair payment and dispute resolution. Deployment is the most gas-intensive, followed by storage-heavy and payment functions.

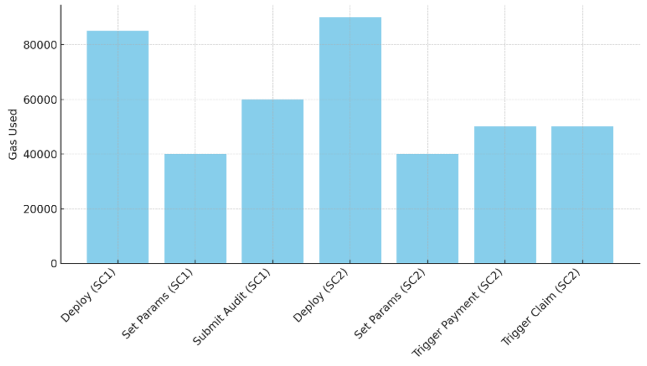

Fig. 11. Smart contract gas consumption.

The core smart contract functions involved in each audit namely, submitting audit evidence and triggering payment consume approximately 90,000 to 100,000 gas in total, which demonstrates the practical feasibility and scalability of the proposed scheme.

F. Comparison with Existing Schemes

Fig. 12 compares our scheme with Zhou et al. and Liu in terms of audit latency and update overhead [10][13]. This scheme consistently outperforms baselines due to:

- Tag aggregation via HLA (vs. per-block tag checking).
- No third-party verifier involvement.
- On-chain automation via smart contracts.

For 1 GB files, our scheme achieves 35 to 50 per cent reduction in audit time and 30 per cent lower update cost.

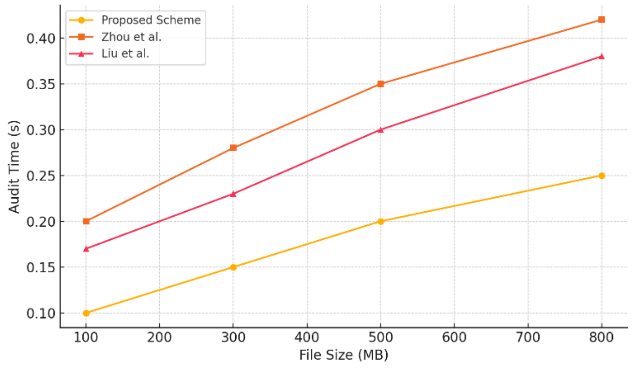

Fig. 12. Comparison with existing schemes.

VII. Conclusion

This study proposes a blockchain-based integrity verification scheme for encrypted cloud data that achieves decentralized auditing, privacy preservation, and fair economic incentives. By integrating Homomorphic Linear Authenticators (HLA), Multi-Party Computation (MPC), and smart contracts, the scheme eliminates reliance on third-party auditors and ensures automatic and transparent dispute resolution and payment settlement between data owners and cloud service providers. The system supports dynamic data operations and enables encrypted data verification without exposing plaintext information. Security analysis confirms that the scheme achieves data integrity, confidentiality, fairness, and resistance to replay and collusion attacks. Experimental results further demonstrate that the proposed protocol achieves low audit latency, acceptable computation and communication overhead, and practical gas consumption.

Despite its advantages, the proposed scheme has several limitations. First, the use of MPC introduces additional communication overhead, which may affect performance in large-scale or resource-constrained environments. Second, the prototype is designed for a single-cloud setting and does not fully consider multi-cloud interoperability or cross-chain audit synchronization. Third, the cost of deploying smart contracts and executing transactions on public blockchains may increase under high network congestion or volatile gas prices.

Future research will focus on three directions:

- Optimizing the MPC protocol and reducing on-chain interaction costs through lightweight cryptographic primitives or zero-knowledge proofs;
- Extending the framework to support multi-cloud and cross-provider integrity verification, enabling collaboration and redundancy among distributed storage services;
- Migrating the scheme to Layer-2 or permissioned blockchain platforms to reduce gas consumption and enhance scalability.

ACKNOWLEDGMENT

The work described in this study was supported by Sichuan Vocational College of Information Technology, Guangyuan, China. Sichuan Vocational College of Information Technology, 2025 Research Project (2025KC02).

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest.

REFERENCES

[1] Mathur P. Cloud computing infrastructure, platforms, and software for scientific research. High Performance Computing in Biomimetics: Modeling, Architecture and Applications. 2024 Mar 21:89-127.

[2] Ali H, Abidin S, Alam M. Auditing of outsourced data in cloud computing: an overview. In2024 11th International conference on computing for sustainable global development (INDIACom) 2024 Feb 28 (pp. 111-117). IEEE.

[3] Zhao Y, Qu Y, Xiang Y, Uddin MP, Peng D, Gao L. A comprehensive survey on edge data integrity verification: Fundamentals and future trends. ACM Computing Surveys. 2024 Oct 7;57(1):1-34.

[4] Shalabi K, Al-Nabhan M, Al Dala'ien MA. Blockchain Based Auditing for Cloud Security: A Systematic Review. InWorld Congress in Computer Science, Computer Engineering & Applied Computing 2025 (pp. 184-199). Springer, Cham.

[5] Hossain MI, Steigner T, Hussain MI, Akther A. Enhancing data integrity and traceability in industry cyber physical systems (ICPS) through Blockchain technology: A comprehensive approach. arXiv preprint arXiv:2405.04837. 2024 May 8.

[6] Perera L, Ranaweera P, Kusaladharma S, Wang S, Liyanage M. A survey on blockchain for dynamic spectrum sharing. IEEE Open Journal of the Communications Society. 2024 Mar 14.

[7] Wang L, Hu M, Jia Z, Guan Z, Chen Z. SStore: an efficient and secure provable data auditing platform for cloud. IEEE Transactions on Information Forensics and Security. 2024 Apr 1.

[8] Sameera KM, Nicolazzo S, Arazzi M, Nocera A, KA RR, Vinod P, Conti M. Privacy-preserving in Blockchain-based Federated Learning systems. Computer Communications. 2024 Apr 20.

[9] Zhang, Y., Geng, H., Su, L., & Lu, L. (2022). A blockchain-based efficient data integrity verification scheme in multi-cloud storage. Ieee Access, 10, 105920-105929.

[10] Zhou, Z., Luo, X., Bai, Y., Wang, X., Liu, F., Liu, G., & Xu, Y. (2022). A Scalable Blockchain-Based Integrity Verification Scheme. Wireless Communications and Mobile Computing, 2022(1), 7830508.

[11] Sankar, S. M., Selvaraj, D., Monica, G. K., & Katiravan, J. (2023). A Secure Third-Party Auditing Scheme Based on Blockchain Technology in Cloud Storage. arXiv preprint arXiv:2304.11848.

[12] Xie, G., Liu, Y., Xin, G., & Yang, Q. (2021). Blockchain-Based Cloud Data Integrity Verification Scheme with High Efficiency. Security and Communication Networks, 2021(1), 9921209.

[13] Liu, Z., Ren, L., Feng, Y., Wang, S., & Wei, J. (2023). Data Integrity Audit Scheme Based on Quad Merkle Tree and Blockchain. IEEE Access, 11, 59263–59273.

[14] Huang, Y., Yu, Y., Li, H., Li, Y., & Tian, A. (2022). Blockchain-based continuous data integrity checking protocol with zero-knowledge privacy protection. Digital Communications and Networks, 8(5), 604-613.

[15] He, K., Huang, C., Shi, J., Hu, X., & Fan, X. (2021, November 1). Enabling Decentralized and Dynamic Data Integrity Verification for Secure Cloud Storage via T-Merkle Hash Tree Based Blockchain. Mobile Information Systems, 2022, 1–17.

[16] Li, J., Wu, J., Jiang, G., & Srikanthan, T. (2021, November). Blockchain-based public auditing for big data in cloud storage. Information Processing & Management, 57(6), 102382.

[17] Wang, M., Zhu, T., Zuo, X., Ye, D., Yu, S., & Zhou, W. (2023). Blockchain-Based Gradient Inversion and Poisoning Defense for Federated Learning. IEEE Internet of Things Journal.