








RESEARCH ARTICLE

WILEY

Trusted audit with untrusted auditors: A decentralized data integrity Crowdauditing approach based on blockchain

Haiwen Chen^{1,2}  | Huan Zhou²  | Jiaping Yu²  |
Kui Wu³  | Fang Liu¹  | Tongqing Zhou²  | Zhiping Cai² 

¹School of Design, Hunan University,
Changsha, Hunan, China

²College of Computer, National
University of Defense Technology,
Changsha, Hunan, China

³Department of Computer Science,
University of Victoria, Victoria,
British Columbia, Canada

Correspondence

Fang Liu, School of Design, Hunan
University, 410082 Changsha, Hunan,
China.

Email: fangli@hnu.edu.cn

Zhiping Cai, College of Computer,
National University of Defense
Technology, 410073 Changsha, Hunan,
China.

Email: zpcai@nudt.edu.cn

Present Address

College of Computer, National University
of Defense Technology, 410073 Chang-
sha, Hunan, China

Funding information

NUDT Research Grants,
Grant/Award Number: ZK19-38;
National Natural Science Foundation of
China (CN), Grant/Award Number:
62072465; National Key Research and
Development Program of China,
Grant/Award Numbers:
2018YFB0204301, 2020YFC2003400

Abstract

Edge computing emerges as an alternative to cloud computing in the scenarios where the end devices require lower latency and faster access speeds. Edge nodes are deployed at the proximity of the end devices to reduce response time. On the other hand, the edge nodes are usually owned by small organizations that have limited operations and maintenance capabilities. Data on the edge may be easily damaged, due to external attacks or internal hardware failures. Therefore, it is essential to verify data integrity in edge computing. However, edge environment requires a different trust model compared with other computing and storage paradigm. Besides, compared with cloud storage, edge storage is decentralized and storage service participants may pose greater internal and external threats. This paper proposes a blockchain-based intelligent crowdsourcing audit approach (Crowdauditing) to achieve on-chain and off-chain credibility of audit results. The model relies on an untrusted auditor committee from the crowd to audit data integrity and uses smart contracts as the core of the intelligent system to ensure the reliability of result submission, the accuracy of the result judgment, and reasonable punishments and rewards. Specifically, an unbiased selection algorithm is proposed to achieve fairness during the auditor committee construction. An

innovative two-stage submission strategy is proposed to ensure that the auditor committee can reach a consensus on the off-chain audit results. An incentive mechanism is carefully designed to force auditors providing audit services honestly to maximize their own rewards. Moreover, we modeled that as a game of n players, which proves the reliability of the result. Finally, we implement a prototype of Crowdauditing based on smart contracts. The extensive experimental results demonstrate the effectiveness of Crowdauditing.

KEYWORDS

crowdauditing, data integrity auditing, edge storage, game theory, smart contract

1 | INTRODUCTION

Cloud computing has great success in the past 10 years, providing users and enterprises with powerful, scalable, and reliable computing and storage. However, the cloud cannot meet the requirements of low-latency and high-speed data access. Edge computing¹ is proposed as a new architecture to solve this problem. It pushes computing power and data storage from the cloud to edge nodes close to terminal devices, so that terminal devices can enjoy low-latency computing and data access services.

With the development of edge computing, the research and development of edge storage solutions has attracted a lot of attention (e.g., Micron Edge Storage of Video-Surveillance,² Centipede solution,³ edge storage of medical data,⁴ and Ctera⁵). Taking the Micron edge storage solution² as an example, portable or mobile devices, such as cameras or handheld tablets, can use secure digital (SD) cards or other storage devices to implement edge storage, and then send locally processed information to the cloud. When retrieving information, different granular information can be retrieved in the cloud or on the edge as needed. The edges usually do not write back the updated data blocks to the back-end cloud immediately, even though the end devices may frequently update the data blocks on the edges. The Micron surveillance video edge storage solution can optimize the design of network load/capacity availability, and redundantly record more detailed data at the edge. In this case, the back-end cloud has no ability to recover the updated data for users if the data on the edges are corrupted, and the users then permanently lose their data. Besides, there are also edge computing solutions that predownload some data and services from the cloud in advance to provide timely data access for end devices. For example, Ctera⁵ improves the user's data access experience by deploying device clusters close to the user side or using the company's existing storage devices to implement cloud-based edge cache storage. Users can enjoy faster data access services by joining nearby edge clusters. Though the data stored in the back-end cloud is intact, the predownloaded data on the edges may also be corrupted. Therefore, data integrity verification, that is, *checking whether the data have been removed or tampered*, is essential for edges.

Although many works have been proposed for integrity verification, previous approaches do not work well in the context of edge computing. Since edge storage not only brings the convenience of high-efficiency storage, access speed, but also poses new challenges in integrity auditing.

1.1 | Challenges

First, edge environment requires a different trust model compared with other computing and storage paradigm. Cloud services are usually provided by a big company acting as a trusted authority, while edge storage services may be provided by different untrusted entities, for example, small companies or even users. In the cloud context, data integrity is taken care of by cloud providers themselves or their selected third-party auditors (TPAs). Taking the data of surveillance video as an example, the data may be stored in the edge servers offered by other entities, which may also work as customers. Hence, it may be difficult to identify a TPA that is acceptable for all entities involved in the edge storage solution.

Second, compared with cloud storage, edge storage is decentralized and storage service participants may pose greater internal and external threats. Storage service participants may maliciously tamper with existing files, and more seriously, delete files, and deceive consumers for various reasons. External threats include node loss or equipment damage due to network reasons or man-made damage. For example, the edge storage device of the video surveillance camera is smashed or the node network is dropped. The higher level of threats requires more frequent data integrity verification. Nevertheless, the traditional cloud TPA model normally assumes a centralized service, which does not fit the high-frequency, distributed data auditing requirements at the decentralized edge.

We are thus motivated to tackle the challenge of auditing data integrity in the untrusted, distributed environment of edge computing. In the context of edge storage systems, the *main goal* of data integrity audit is to achieve:

- *Decentralized auditors*: To avoid the problem of using centralized TPA, we need to build a decentralized audit model, where an auditor committee is dynamically formed by multiple auditors in an untrusted environment to release the heavy burden from a single centralized auditor.
- *Edge storage correctness*: It can audit whether the data stored by the Edge exists or has been tampered with, and ensure that the Edge cannot cheat to pass the audit.
- *Trustworthy auditing result*: We need to ensure that the audit results are credible. The audit results need to be audited by the auditor rather than fabricated, and the results cannot be tampered with or plagiarized in the process of transmitting the results.

We achieve all the above goals by developing *Crowdauditing*, an intelligent data integrity verification model for general-purpose data storage services in edge computing. The Crowdauditing, utilizing homomorphic verifiable tags (HVT),⁶ hands over the audit task to an auditor committee as a crowdsourcing task to achieve intelligent decentralized audit.

Besides, blockchain technology⁷ becomes a natural solution to ensure that the stored data are immutable and is widely used in the field of data recording and management.⁸ The smart contracts in Ethereum⁹ provide a feasible way to automate the audit transactions and enforce the judgment. It is natural for us to think of using blockchain technology as the solution.

Nevertheless, data may be manipulated before being stored in the blockchain. We thus need to improve the tamper resistance of the off-chain events' results. The design of smart contracts, as an intelligent system that achieves the reliability of results through the design of result submission, judgment of results and punishment mechanism, ensure that auditors have to tell the truth and punish lying or lazy nodes, which is the core content of the Crowdauditing.

Furthermore, to further illustrate the rationality of smart contracts, we introduce game theory¹⁰ to analyze the reliability of the results submitted by all auditors and model the results submitted by all auditors as a game of n players. Game theory is one of the formal studies of decision-making and deals with the problem of conflict and cooperation.^{10–13} In a game, players will always play their best strategies to maximize their payoffs. Payoffs are values indicating how big the rewards a player will get if it applies its strategies. When the benefits of all players in a game are maximized, the game reaches the Nash equilibrium.¹⁴ Through detailed game analysis and the analysis of the Nash equilibrium point, we can know that the auditor has to tell the truth to maximize the benefits, so as to finally achieve the reliability of all audit results.

The main contributions of this paper are summarized as follows:

- We propose a blockchain-based intelligent data integrity audit framework for general decentralized edge storage based on smart contracts, and propose an unbiased randomized selection mechanism to compose a pragmatic auditing committee composed of decentralized and untrusted auditors from the crowd for data integrity auditing in edge computing.
- We propose a two-stage submission mechanism based on smart contracts, which can effectively prevent tampering and plagiarism among results submitted by auditors in committee. It further empowers our model with the ability of decentralized auditing.
- We design an incentive mechanism to prompt the untrusted auditors to report results truthfully. Besides, we model the results submitted by all auditors as a game and obtain the Nash equilibrium point. The Nash equilibrium principle of game theory is leveraged in our model to ensure trustworthiness.
- We implement the *Crowdauditing** model based on the worldwide public blockchain test net “Rinkby”¹⁵ of Ethereum. The implementation and evaluation demonstrate the feasibility and the performance of our *Crowdauditing* model.

1.2 | Organization

The rest of the paper is organized as follows. We discuss the related work in Section 2. Section 3 describes the background and threat model. Section 4 presents the decentralized *Crowdauditing* model. Our key techniques and detailed design are described in Section 5. Section 6 analysis the security of *Crowdauditing*. Section 7 introduces the prototype implementation and experiments. Finally, Section 8 concludes the paper.

2 | RELATED WORK

2.1 | Traditional verification

There are mainly two types of traditional data integrity verification mechanisms. One is Provable Data Possession (PDP), and the other is Proof of Retrievability (POR).¹⁶ PDP can quickly verify

whether the data stored on the cloud is intact, while POR can restore the damaged data when the data integrity is compromised. Deswarte et al.¹⁷ proposed the basic PDP authentication method. Juels and Kaliski¹⁸ proposed the notion of POR, which relies on indistinguishable blocks to detect data corruption without using the public auditing model. Public verification with the TPA better supports the characteristics of the dynamic update and efficient verification. Hao et al.¹⁹ and Liu et al.²⁰ proposed signature-based HVTs technology and homomorphic hash technology to verify data integrity. Li et al.²¹ proposed a mechanism name “OPoR,” which outsources and offloads the heavy computation of the tag generation to the cloud audit server and eliminates the involvement of users in the auditing and in the preprocessing phases. The research²² optimized an existing third-party auditing protocol and make it resistant to replace, replay- and forge attacks launched by malicious insiders at cloud storage server. However, the above work assumes that TPA is honest or semi-honest, this is obviously undesirable under different trust models in edge computing scenarios.

In practice, In the traditional scenario where the TPA is introduced for data integrity verification, TPA may collude with Clients or Providers to forge results,²³ and some studies try to solve the untrustworthy problem of TPA. Huang et al.²³ proposed to use multiple TPAs for audit, based on the assumption that there at least exist some trusted auditors. Worku et al.²⁴ also proposed a solution, but it is vulnerable to forgeries using known message attacks from the malicious service provider.^{25,26} Overall, these solutions use some complex computing modes and thus increase the computing overhead of the storage nodes or TPAs.

2.2 | Auditing with blockchain

Blockchain technology implements decentralized peer-to-peer transactions, coordination, and collaboration without the need for trust, through data encryption, time stamping, and distributed consensus. It can address the problem of high cost, inefficiency, and insecure data storage of centralized systems. The trust problem brought by traditional data integrity verification makes it an inevitable trend to integrate blockchain⁷ into data integrity verification. Liu et al.²⁷ proposed to apply blockchain to avoid using TPA, and Yue et al.²⁸ proposed a blockchain-based framework trying to achieve trustworthy audit results. Both of them lack the necessary considerations for ensuring the trustworthiness of the off-chain events' outcome. Hao et al.²⁹ realized the data verification based on a private blockchain in an untrusted environment, but their solution needs to construct and deploy a private blockchain, which is very difficult in practice. Zhou et al.³⁰ proposed a witness model to credibly enforce the off-chain cloud Service Level Agreement (SLA) based on smart contracts. Their work is aimed at SLA and does not apply to storage audits for edge computing and the result submission may be delayed. Miao et al.³¹ proposed a mechanism to use block Hash to generate Challenges, but this method has no guarantee that the audit results cannot be tampered with off-chain. There are also some multi-auditor models based on the blockchain^{29,32}; however, their proof verifying process is in the smart contract or using proof of work in blockchain, which consumes excessive fees of the public blockchain or uses more verifying time. Besides, they developed their own chain, which is hard to promote in practice.

2.3 | Auditing in edge computing

Recently, some research has aimed at auditing the integrity of data in edge computing.^{33–37} Wang et al.³³ researched the balance between privacy and data integrity in edge-assisted

internet of things, but they did not solve the issues raised in our challenges. Tong et al.³⁴ proposed to check data integrity on a single edge or multiple edges. Liu et al.³⁵ proposed an efficient data integrity auditing in edge computing, however, their work is still based on the existence of trusted TPA. Li et al.³⁶ implemented the work of App vendor to challenge the integrity of cache data stored at the edge nodes, but it is only used for edge caching. Recently, Yue et al.³⁷ proposed a decentralized audit solution for data integrity using blockchain in the edge computing scenario, but their “ProofVerify” work is placed on the blockchain, which increases the overhead on the chain. Besides, users in their scheme need to keep connecting to the edge for challenging the integrity of the data, which may increase the burden on end-users.

3 | BACKGROUND AND THREAT MODEL

Background: We focus on data auditing in edge computing scenarios as shown in Figure 1. There are mainly four roles in the scenarios: Data Owner (DO), Edge, Auditor Committee (AC), and Service Provider (SP).

1. *Data Owner DO*: requires distributed storage of its encrypted data, and then pays for each round of storage auditors.
2. *Storage Provider SP*: servers as a storage SP, provides users with services and system management to obtain profits.
3. *Edge*: provides storage spaces to obtain profits.
4. *Auditor Committee AC*: who faithfully challenges the Edges, validates the response proof in each round of the storage auditing protocol.

The *Edge* provides storage services for *DOs*. During the service time T_{service} , *SP* needs to ensure that data cannot be deleted or tampered with. *AC* challenges *Edge* to check the integrity of the data whenever needed. When T_{service} ends, if data are consistent and complete, the *DO*

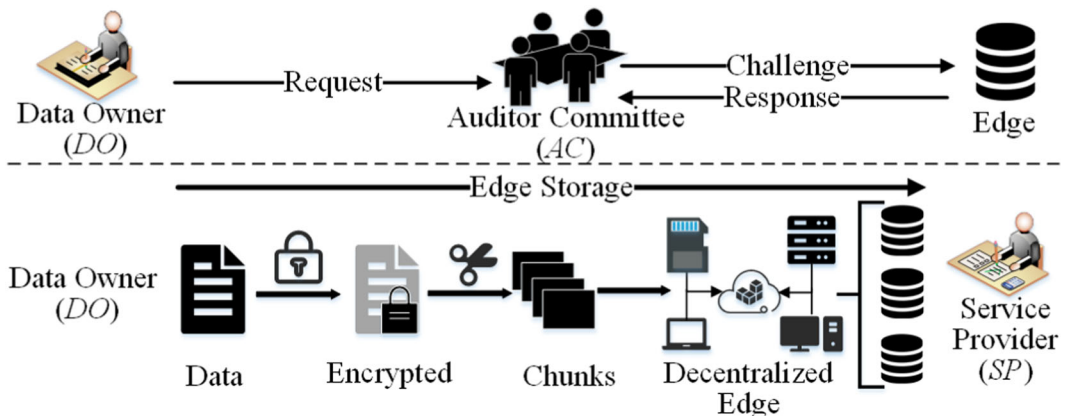


FIGURE 1 Edge storage and the auditing model based on Auditor Committee. AC, Auditor Committee; DO, Data Owner; SP, Service Provider [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/m.2248)]

needs to pay the SP Fee_{service} for data storage. Otherwise, the SP will not get any benefits and even needs to compensate the DO for $Fee_{\text{compensation}}$.

For simplicity, we assume that all *Edges* constitute the overall edge storage service and we emphasize that encryption is a mandatory measure taken by the DO in this paper. The storage provider, that is, the SP , participates in the service as an administrator. The DO divides all the data into chunks and stores the chunks in multiple *Edges*. When estimating the actual cost, the DO can freely adjust the data redundancy level and the number of chunks, which may lead to a linear increase in the total audit cost.

Threat model: In this paper, we need to consider the credibility of all parties:

- *Auditors in AC behave lazily or conspiracy:* Auditors may lazily report beneficial results without auditing. Besides, auditors in AC may also collude to report consistent results to maximize their incentives.
- *Edge's behavior is complicit:* *Edge* may simply delete data to reclaim more storage space for more monetary benefits or subtly discard data that DO rarely accesses to save bandwidth; considering the reputation, TPA may also hide data loss events.
- *DO's behavior is complicit:* In extreme cases, DO may generate incorrect verification information for *Edge*, so that the auditor will always maliciously falsely claim that the data are lost to save costs for DO .

Our model in this paper addresses the above threats, and we conduct a detailed analysis in Section 6.

4 | OUR CROWDAUDITING MODEL

To tackle the trust issues of the single TPA mechanism and ensure that the audit task is carried out, we propose the *Crowdauditing* model via the smart contract of public blockchain as the trusted platform to replace the single TPA audit mechanism and determine whether the data are abnormal.

The specific audit process is shown in Figure 2. First, the DO and the SP negotiate to determine the details, the employed auditors' number, service fees that can be paid, and some other information. After the *Edges* store the data in the system, the SP publishes the audit requirements to the smart contract according to the negotiation with the DO . Any participant of our system, that is, any blockchain user willing to join the audit event, constitutes a crowd, and the auditors from the crowd can constitute the (AC) to perform a task by paying a deposit. In the scenario of edge storage, users who wish to store and nodes that provide storage services constitute a huge group. Therefore, we assume that the attacker's resources are limited. It is impractical for attackers to create a large number of users with disguised identities to increase the possibility of being selected from the crowd to enter the AC , which can be traditional and affect the audit results. After AC is formed, the AC members then begin the audit tasks based on the details.

As can be seen from Figure 2, our *Crowdauditing* model mainly consists of two smart contracts as follows.

Crowd manage smart contract (CMSC): Any participant of our system, that is, any blockchain user willing to join the audit event, constitutes a crowd, and the auditors from the crowd can constitute AC to perform a task by paying a deposit. The identity of each auditor

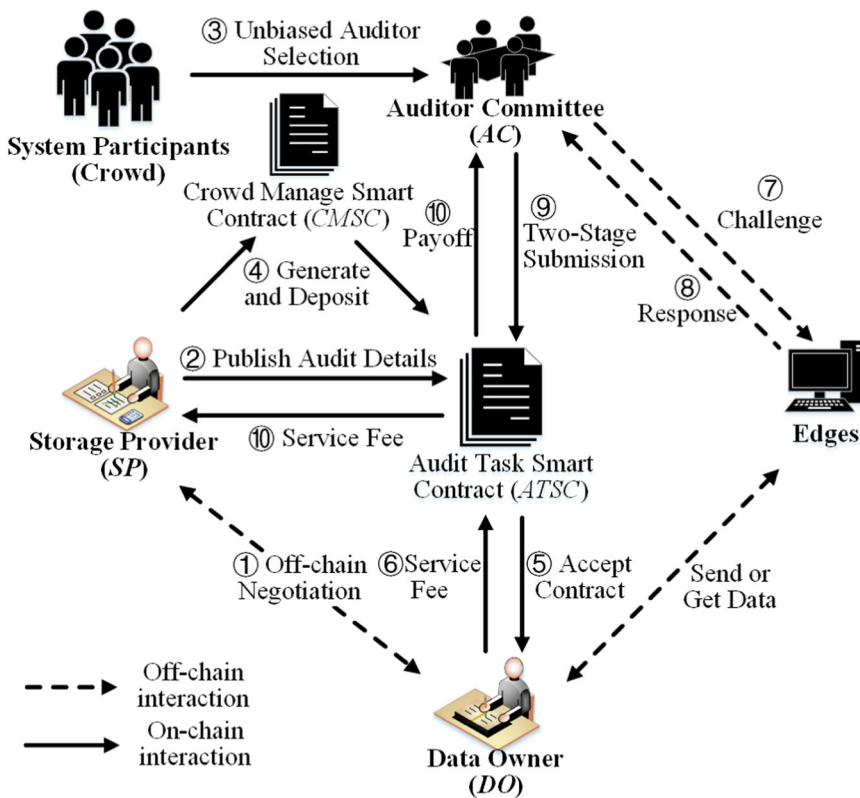


FIGURE 2 Decentralized Crowdauditing based on smart contracts [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/m2.2348)]

is just its wallet address in the blockchain network. Besides, any user cannot audit data related to himself/herself, that is, data stored in his/her own node or data belonging to himself/herself. Members with negative reputations cannot join the auditor committee.

Audit task smart contract (ATSC): The ATSC determines whether the data are integrated through the results reported by the auditors in the AC. It is worth mentioning that we assume that all AC participants are independent with each other, even if collusion is involved, the method of unbiased auditor selection can ensure that most of the selected auditors cannot be predetermined for colluding. Besides, ATSC is responsible for the decision-making of the final result, and rewards or punishes auditors in AC based on the decision. At the same time, the auditor's reputation is updated according to the decision results, and the updated reputation value is also the basis for whether to join the crowd.

We stress that the auditing frequency should be set properly (e.g., every few hours), such that compared to the auditing interval, the time of block confirmation is much smaller. We take the Mean Time Between Failure (MTBF)³⁸ as the audit window, for example, the average offline time of nodes in the system and the average time of abnormal detection. In an MTBF window, AC can implement multiple rounds of audit tasks, and the frequency of the audit can be negotiated and determined by the DO and the SP. Each auditor in AC can challenge and verify whether the data are abnormal in the time window. In the end, each auditor submits

their audit result to the smart contract independently. When all the audit results are received correctly, each participant can withdraw its corresponding rewards.

The *Crowdauditing* model relies on the following five critical components, which are what we mainly describe in detail in Section 5:

- *Unbiased auditor selection*: The selection of auditors from the crowd for constructing the AC should be unbiased. The auditors cannot audit the data stored on their own nodes or the data published by themselves. By deploying the unbiased auditor selection on CMSC, we get the AC consists of n auditors, who are selected from the crowd in an unbiased manner.
- *Details of integrity auditing*: We utilize a basic HVT-based method to achieve trustworthy integrity verification results. Specifically, in the phase of proof verification, we outsource the proof work to AC to obtain credible results through decentralized verification.
- *Two-stage submission strategy*: Both of the above components are involved in submitting information to the Blockchain. The two-stage submission strategy achieves the purpose of pseudo-simultaneous submission by requiring the submission of commitments within a certain period of time, thereby preventing them from plagiarizing each other's reports. In our model, the strategy is applied to enhance the reliability of unbiased auditor selection and audit results submission.
- *Payoff function*: The payoff function deploying in the ATSC incentivizes the AC members, who are rational and greedy, to provide honest audit results to seek their own maximum interests, which is fully proved through game theory.
- *Auditors' reputation*: In the CMSC smart contract, to eliminate some malicious auditors from joining the AC, the reputation of the blockchain is used for auditor control instead of feedback from other nodes.

5 | KEY TECHNIQUES AND PAYOFF FUNCTION DESIGN

In this section, we mainly describe the four components mentioned above in detail. And, in the end, we use a game theory-based method to show that AC members must act honestly and tell the truth to maximize their profits through the Nash equilibrium principle.

5.1 | Unbiased auditor selection

In the *Crowdauditing*, how to select auditors from the Crowd is a crucial problem. The selection of committee members must be unbiased. DOs and SPs cannot become supervisors of related data and cannot have preferential treatment in the process of selecting members. In this paper, we design an unbiased auditor selection mechanism based on the smart contract of Ethereum to form a trustworthy auditor committee.

In Ethereum, we use a CMSC to implement member management of the Crowd, and any member of the blockchain can join the Crowd by registering. The CMSC manages the addresses of the Crowd members as a list. The Crowd members can join or exit the Crowd by modifying the status "Online" or "Offline." The specific implementation process and interface design of the crowd management smart contract are shown in Figure 7.

The main function of CMSC is an unbiased selection of auditors from the crowd to form an AC. When the SP issues an audit task, it will first submit the parameters to CMSC. The

parameters include the required number of auditors and a random number. The *DO* will also provide a random number to *CMSC*. After *CMSC* finishes receiving the random number, another interface “startSelected” is called to use two random numbers as a seed to complete the selection process. The selection algorithm is shown in Algorithm 1.

Algorithm 1. Unbiased auditor selection

Input:

Registered auditor address list: RA ;
 Required number of committee member: N ;
 Random hash provide by Provider: $Rand_p$;
 Random hash provide by Data Owner: $Rand_o$;
 The address of Provider: $Addr_p$;
 The address of Data Owner: $Addr_o$;

Output:

Selected auditor address list of committee: SA ;
 1: assert ($len(RA) \geq 10*N$);
 2: $seed = Hash(Rand_p + Rand_o)$;
 3: $SA = \emptyset$;
 4: $j = 1$;
 5: **while** $j \leq N$ **do**
 6: $index = seed \% len(RA)$;
 7: **if** $RA[index].state = Online$ and $RA[index].reputation > 0$ and $RA[index].address \neq Addr_p$ and $RA[index].address \neq Addr_o$ and $RA[index].address \notin SA$ **then**
 8: $RA[index].state = Await$;
 9: $SA \leftarrow RA[index]$;
 10: $j++$;
 11: **end if**
 12: $seed = Hash(seed)$;
 13: **end while**
 14: Return SA ;

In Algorithm 1, two random numbers $Rand_p$ and $Rand_o$ provided by *SP* and *DO* are used as the seeds of the committee members' selection. After each member is selected, the seed is hashed again as the basis for the next round of selection, until all required auditors of *AC* are selected. They hence can be convinced that the selected committee members are not dominated by the opponent or any other third party. Because they both provide part of the randomness.

Although *SPs* or *DOs* can use the form of registering multiple addresses in the crowd to represent their own interests, if there are enough registered auditors in the Crowd, the chance of selecting most of its abusive members at the same time is still very small. Therefore, considering the above problems, at the beginning of the algorithm, we ensured that the number of members in the Crowd is much greater than the required number of committee members by setting the assert restrictions as 10 times greater.

Besides, given that the data in the blockchain are visible to all users when submitting random numbers, the submitter may have the possibility of plagiarism due to the submission order, leading to the possibility that one party may influence the selection of audit nodes in this

way. Therefore, we realize the fairness of random number submission through the two-stage submission described in Section 5.3.

On the basis of the above mechanisms, we can guarantee that the selection of committee members is unbiased and cannot be manipulated by relevant stakeholders, which are *SP*, *DO*, or auditors in the *AC*.

5.2 | Details of integrity audit

Denote the file to be stored as *F*. It is further encrypted and divided into *n* data blocks in the form of group elements. Then, each *s* collection of data blocks can constitute data chunks for the acceleration of data processing and the savings of extra storage overhead.

- (1) *Setup*: The setup stage mainly includes the following steps:
 1. *DO* generates G_1, G_2, e , where G_1 and G_2 are two finite cyclic groups of large prime p . g is a generator of G_1 . $e: G_1 \times G_1 \rightarrow G_2$ is a Bilinear Map.³⁹ $H: \{0, 1\}^* \rightarrow G_1$ is a hash function.
 2. *DO* selects a random number $a \in \mathbb{Z}_p^*$, computes the public value g^a , sets the public key $pubKey = g^a$ and the private key $priKey = a$.
 3. The parameter u is a random value in G_1 . The public parameters $pp = \{G_1, G_2, p, e, g, u, g^a, H\}$.
- (2) *SigGen*: Before the data are sent to the *Edge*, *DO* should generate the signature used in *AC*.
 1. The *DO* divides the file *F* into *n* blocks, the set of all blocks is $B = (b_1, b_2, \dots, b_n)$, where $b_i \in \mathbb{Z}_p$.
 2. *DO* needs to sample a file identifier called *name* from \mathbb{Z}_p^* such that $H(name \parallel i)$ can be used for block indexing, $0 < i \leq n$.
 3. For each block b_i , the *DO* calculates the corresponding signature $\sigma_i = (H(name \parallel i) \cdot u^{b_i})^{priKey}$, where $H(name \parallel i)$ is the identifier of b_i , the signature can be denoted as a set $\Phi = \{\sigma_i\}$, where $0 < i \leq n$.
 4. The *DO* sets $\tau_0 = name \parallel g^a$ and calculates the file tag by computing $\tau = \tau_0 \parallel TagSig(\tau_0)$, where $TagSig(\tau_0)$ is the signature on τ_0 under the signing private key *ssk*.

After generating the signature, *DO* sends the $\{F, \Phi\}$ to *Edges*, sends τ to auditor in *AC*, and then could delete the data and the signature from the local storage.

- (3) *Challenge*: To verify the data integrity, the auditor in the *AC* only needs to randomly select a certain number of blocks instead of all the blocks. This probabilistic method can significantly improve the efficiency with the guarantee of high detection probability of the polluted blocks.⁶ The challenge message is generated as follows:
 1. Randomly select a subset *I* with *c* blocks from *B*: $S = \{s_1, s_2, \dots, s_c\}$ which will be challenged. For $\forall i, j \in [1, 2, \dots, c]$, $s_i \neq s_j$ if $i \neq j$.
 2. $\forall i \in I$, select a random number $v_i \in \mathbb{Z}_p^*$.
 3. Denote the message of a challenge $chal = \{i, v_i\}, i \in I$.

When challenging a File *F*, the auditors in *AC* initiates the challenge by sending *chal* to the *Edge*.

- (4) *ProofGen*: Upon receiving the random challenge *chal* from auditors in *AC*, the *Edges* generate a proof of the data as follows:
1. The *Edge* computes the linear combination of challenged blocks: $\mu = \sum_{i \in I} b_i v_i \in \mathcal{Z}_p^*$, $\mu_i = b_i v_i$.
 2. The *Edge* computes an aggregated signature, $\sigma = \prod_{i \in I} \sigma_i^{v_i} \in G_1$.
 3. Output an auditing proof $proof = \{\mu, \sigma\}$ and respond to the auditors in *AC*.
- (5) *ProofVerify*: On receiving *proof* from *Edge*, the auditors in *AC* verifies the correctness of auditing proof as follows:

$$e(\sigma, g) = e\left(\prod_{i \in I} H(name \mid i)^{v_i} \cdot u^\mu, g^a\right). \quad (1)$$

If (1) holds, the file stored in the *Edge* is intact; otherwise, it is not.

5.3 | Two-stage submission strategy

After the n ($n \in \mathbb{N}^+$) auditors in the committee has audited the data t times within an MTBF³⁸ time window for an audit task T , each auditor A_k ($0 < k \leq n$ and $k \in \mathbb{N}^+$) in the committee gets the result $R_k = \{r_{k1}, r_{k2}, \dots, r_{kt}\}$ and the results of all auditors form a result matrix $R_{Matrix} = (R_1, R_2, \dots, R_n)^T$. To avoid tampering during submit R_{Matrix} , we propose to use a two-stage submission strategy based on the smart contract. The results submission mechanism including the commitment submission stage and the result reveal stage.

The first stage in Figure 3 is the commitment submission stage. Each Auditor _{k} submits its own commitment C_k , $C_k = H(R_k, S_k)$, where S_k is the random number generated by the Auditor _{k} , $S_k \in \mathbb{Z}_p^*$, R_k is the audit results of Auditor _{k} and H is a secret hash function. In this stage, the value of S_k is in the state of confidentiality, to prevent other auditors from inferring the result value based on the value of S_k . Besides, if two different auditors submit the same commitment, $C_a = C_b$, ($0 < a, b \leq n$ and $a, b \in \mathbb{N}^+$), *ATSC* will also classify the situation as plagiarism, and the later submitted auditor will be rejected. Then, the auditor should reselect the random secret value S_k and resubmit. Through the above design, auditors in the committee can make sure that everyone has gotten their results and submitted a commitment, but they cannot know the detailed results of others. Therefore, by promising to deposit evidence, committee members can be confident that others have completed their tasks and that the unpublished result data cannot be tampered with later on. Within the specified time window, if the auditor in the committee fails to upload the commitment deposit, it is deemed that the audit task has not been completed. Then, the state of audit turns to a failure state. The contract calls the *resetAudit* method to reset the audit task and deducts the deposit of auditors in *AC* who fail to submit the results as a penalty.

After all *AC* members submit commitments on time, Auditor _{k} calls the *Submit* method to upload the results R_k and the random secret value S_k to the *ATSC*. Then, the *ATSC* judges the correctness of results through the commitments submitted in the first stage, that is, judge whether the three values of R_k , S_k , and C_k satisfy $C_k = H(R_k, S_k)$, and only the results meeting the condition can be accepted by *ATSC*. The design ensures that each auditor cannot modify the result data value he/she promised to submit at this stage, otherwise the verification cannot pass. Hence, the possibility of cheating is eliminated.

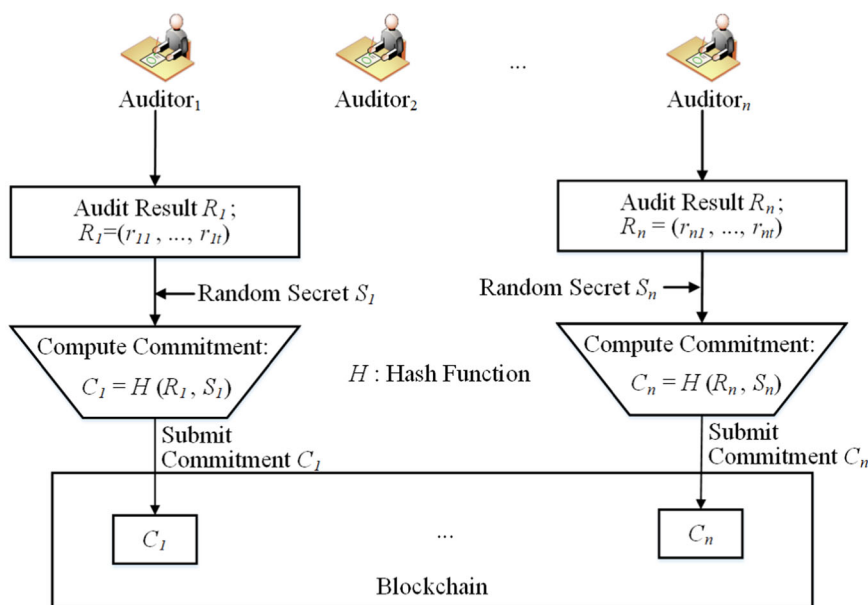


FIGURE 3 The commitment submission stage [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/m.22348)]

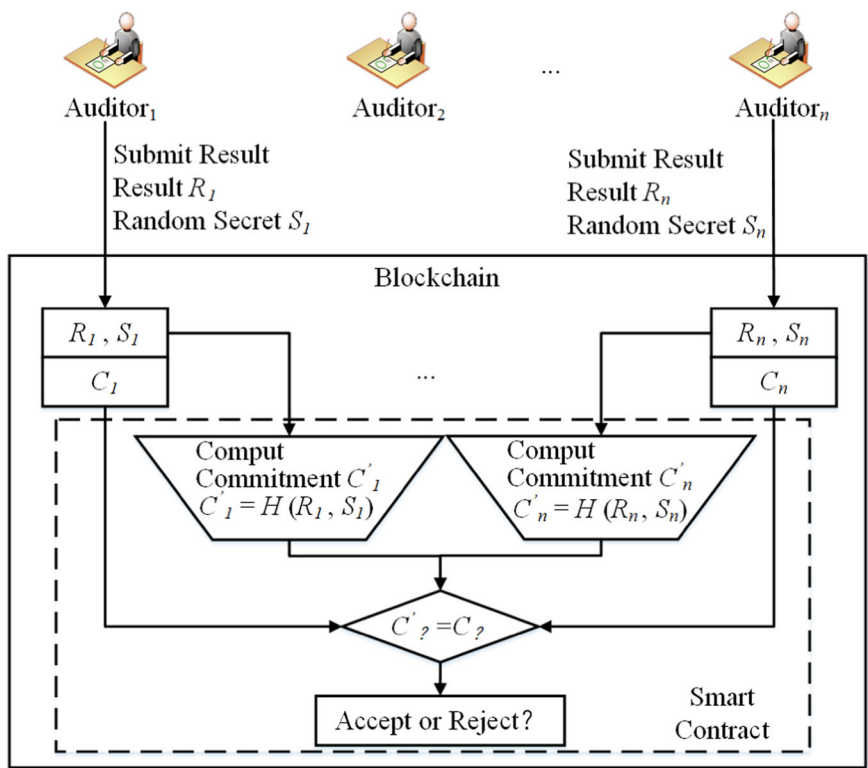


FIGURE 4 The result submission stage [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/m.22348)]

When all *AC* auditors submit effective audit results, the *ATSC* is triggered to evaluate the results, which compares all the results provided by all auditors. The final judgment of the data integrity in each MTBF time window is jointly determined if more than half of the audit results are the same. In this way, we obtained the results of multiple rounds of audits in different MTBF time windows.

5.4 | Payoff function and Nash equilibrium

After the results from all the auditors are evaluated, the incentives need to be allocated to the auditors for rewards. *ATSC* provides embedded salary distribution rules, namely, the payoff function. The design of the payoff function follows the rule that the result provided by each auditor needs to be the same as the result of most of the other auditors to obtain the reward, otherwise, that portion of reward cannot be obtained.

The results pass the verification of the second stage in the two-stage submission strategy are accepted, which can participate in the distribution of incentives. Multiple failed submissions of results should reduce its credibility from the perspective of reputation. Auditors whose reputation is less than 0 or less than a certain value cannot participate in future audit tasks.

To explain the incentive function conveniently, we use the first-round result of each auditor submitted, that is, the first column in the result matrix $R_{\text{Matrix}} = (R_1, R_2, \dots, R_n)^T$. We model the *AC*'s revenue problem as an n -player game Γ . The game consists of a group of participants, strategies, and an incentive function. According to the basic types of complete information strategy games in game theory, the Audit Game is defined as follows.

Definition 1 (Game Γ). It is an n -player game represented as a triple (AC, Σ, Π) , where

- $AC = \{A_1, A_2, \dots, A_n\}$ ($n > 3, n \in \mathbb{N}^+$) is a set of n players. Each player is an auditor in *AC*.
- $\Sigma = \{\Sigma_1, \Sigma_2, \dots, \Sigma_n\}$ is the strategy set of n -auditors, where $\Sigma_k \in \Sigma$ are the sets of strategies for A_k . A_k can choose any actions $\sigma_k \in \Sigma_k$. A strategy profile is therefore a vector, $\Sigma, \sigma^* = \{\sigma_1^*, \sigma_2^*, \dots, \sigma_n^*\}$, where σ_k^* is a specific action of Σ_k .
- $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ is a set of payoff functions, where $\pi_k: \Sigma \rightarrow P$ is the payoff function determining the revenue for auditors A_k in *AC* under a certain strategy, $k = (1, 2, \dots, n)$. P is the revenue.

Besides, for the two-stage n -player game, $\sigma_{-k} = \{\sigma_1, \dots, \sigma_{k-1}, \sigma_{k+1}, \dots, \sigma_n\}$ is defined as any strategy profile σ without player k 's action. The full strategy can then be written as $\{\sigma_k; \sigma_{-k}\}$.

Definition 2 (Actions). Considering that the *AC* members are noncolluding, there are two types of actions for each auditor in general: A_k reports data are normal, that is, not damaged, which is denoted as σ_k^{normal} ; otherwise, A_k reports the data are abnormal, that is, damaged, which is denoted as $\sigma_k^{\text{abnormal}}$.

For the results submitted, we define the set of auditors who report the normal results as A_{normal} , where $A_k \in A_{\text{normal}}$, the $\sigma_k^* = \sigma_k^{\text{normal}}$. Respectively, A_{abnormal} is the set of abnormal results, where $A_k \in A_{\text{abnormal}}$, the $\sigma_k^* = \sigma_k^{\text{abnormal}}$.

Only when most auditors successfully submit the commitment, we can get the committed results and decide whether the file is normal (i.e., integrated) or abnormal (i.e., damaged) by the σ^* as defined in Definition 3.

Definition 3 (Decisions). For the actions from n -players, $\sigma^* = (\sigma_1^*, \sigma_2^*, \dots, \sigma_n^*)$. Only when $\|A_{\text{response}}\| \geq K$, the number of auditors submitted is valid and we get the result of the corresponding submission after that. Only when $\|A_{\text{normal}}\| \geq M$ we confirm the data are integrated, $Data_{\text{Status}} = \text{Integrity}$, otherwise, $Data_{\text{Status}} = \text{Damaged}$, where $1 < N/2 < M < N$, $k, K, M, N \in \mathbb{N}^+$, otherwise, data are lost or corrupted.

Once the result reveals stage is completed or the deadline has expired, ATSC rewards or punishes the auditors according to payoff functions as defined in Definition 4.

Definition 4 (Reward). The incentive and penalty rules can be defined as follows:

- When $Data_{\text{Status}} = \text{Damaged}$:
 - $\forall A_k \in AC, \pi_k(\sigma_k^{\text{normal}}) = -2$;
 - $\forall A_k \in AC, \pi_k(\sigma_k^{\text{abnormal}}) = 5$.
- When $Data_{\text{Status}} = \text{Integrity}$:
 - $\forall A_k \in AC, \pi_k(\sigma_k^{\text{normal}}) = 1$;
 - $\forall A_k \in AC, \pi_k(\sigma_k^{\text{abnormal}}) = -1$.

A Nash equilibrium point⁴⁰ can be defined as a stable state in which no participant is motivated to deviate from the current strategy. When the state satisfies the conditions for each participant to achieve their maximum benefit, the auditor in AC has no other better strategy to replace, and the system reaches a stable state.

Definition 5 (Nash equilibrium point). For A_k in the AC , we can get the best strategy profile $\sigma_k^* \in \Sigma_k$ that achieve the best reward for every auditor $\pi(\sigma_k^*) \geq \pi(\sigma_k)$, $\sigma_k \in \Sigma_k$, $k = (1, 2, \dots, n)$. Therefore, the specific strategy profile σ^* is the Nash equilibrium point.

In the n -player game, there are two dynamic stabilities, that is, when all nodes report that the data are in a normal state or an abnormal state. According to Definition 4 and the definition of the Nash equilibrium point in Definition 5, we can prove that all auditors have the maximum benefit. The mixed strategy of audit results that is not in these two states has not reached the state of Nash equilibrium.

Theorem 1. *In the n -player of ATSC, there are two Nash equilibrium points:*

- $\sigma^* = (\sigma_1^{\text{normal}}, \sigma_2^{\text{normal}}, \dots, \sigma_n^{\text{normal}})$, of which $\forall A_k \in AC$.
- $\sigma^* = (\sigma_1^{\text{abnormal}}, \sigma_2^{\text{abnormal}}, \dots, \sigma_n^{\text{abnormal}})$, of which $\forall A_k \in AC$.

Proof. According to Definitions 1–3 in an N -player game, $N \geq 3$, $N/2 < M \leq N - 1$, and $M \geq 2$, where $M, N \in \mathbb{N}^+$.

For the strategy profile of $\forall A_k \in AC, \sigma_k^* = \sigma_k^{\text{normal}}$ means that $\|A_{\text{normal}}\| = N > M$. The data file is therefore complete. On the basis of the Reward function in Definition 4. For $\forall A_k$, the reward is $\pi_k(\sigma_k) = \pi_k(\sigma_k^{\text{normal}}) = 5$. If any A_k chooses the other action “abnormal” instead of “normal.” The final status of the result, also, would not be modified, due to $\|A_{\text{normal}}\| = N - 1 \geq M$, then, the

reward is $\pi_k(\sigma_k) = \pi_k(\sigma_k^{\text{abnormal}}) = 0$. Then according to Definition 5, this strategy profile is a Nash equilibrium point.

Analogously, for the strategy profile of $\forall A_k \in AC, \sigma_k^* = \sigma_k^{\text{abnormal}}$ means that $\|A_{\text{abnormal}}\| = N > M$. Therefore, the data file is incomplete or tampered with. On the basis of the Reward function in Definition 4. For $\forall A_k$, the reward is $\pi_k(\sigma_k) = \pi_k(\sigma_k^{\text{abnormal}}) = 5$. If any A_k chooses the other action “normal” instead of “abnormal.” The final status of the result, also, would not be modified, due to $\|A_{\text{abnormal}}\| = N - 1 \geq M$, then, the reward is $\pi_k(\sigma_k) = \pi_k(\sigma_k^{\text{normal}}) = 0$. Then, according to Definition 5, this strategy profile is a Nash equilibrium point.

For all the other strategy profiles, they are all a mix of actions or not the best strategy. If all auditors keep response, the other strategy profiles for the results are mixed, it means $\|A_{\text{normal}}\| \neq \emptyset$ and $\|A_{\text{abnormal}}\| \neq \emptyset$ always. If $\|A_{\text{normal}}\| \geq M$, there always $\exists A_k \in A_{\text{abnormal}}$, it can change the status from “abnormal” to “normal,” then $\|A_{\text{normal}}\| + 1 > M$, it will not change the final result. Therefore, A_k increases its reward from $\pi_k(\sigma_k^{\text{abnormal}}) = 0$ to $\pi_k(\sigma_k^{\text{normal}}) = 5$. Analogously, if $\|A_{\text{abnormal}}\| \geq M$, there always $\exists A_k \in A_{\text{normal}}$, it can change the status from “normal” to “abnormal,” then $\|A_{\text{abnormal}}\| + 1 > M$, it cannot change the final result. Therefore, A_k increases its reward from $\pi_k(\sigma_k^{\text{normal}}) = 0$ to $\pi_k(\sigma_k^{\text{abnormal}}) = 5$.

According to the above proof of the incentives function, it can be known that all auditors should provide the same result to reach the Nash equilibrium point. Considering that the auditors participating in the game are independent of each other, which is guaranteed by unbiased random selection, all auditors must faithfully complete their audit tasks to maximize their incentives, that is, to achieve Nash equilibrium. Therefore, auditors must be honest to maximize their income.

We take the 3-auditors game as an example to illustrate the specific game process of the above game theory game. According to Definition 3, N can only take the value 2, and the corresponding $K \geq 2$. According to Definitions 2 and 4, we can obtain all the strategy combinations of 3-players and all the rewards under each combination as shown in Table 1. Obviously, the Nash equilibrium points of the 3-player example are (5, 5, 5) and (1, 1, 1).

Through the above examples, we can also confirm the correctness of our theoretical analysis. As an independent individual, each auditor will make strategic choices to maximizing his revenue. The established benefit acquisition mechanism forces every auditor to make such a choice. When the data are abnormal, every auditor knows that most auditors will report the data abnormality. Only honestly reporting the abnormal state of the data can maximize the benefits. More revenue push auditors to tell the truth. This reaches a Nash equilibrium point $(\sigma_1^{\text{abnormal}}, \sigma_2^{\text{abnormal}}, \sigma_3^{\text{abnormal}})$. Analogously, when the data are integrated, another Nash equilibrium point $(\sigma_1^{\text{normal}}, \sigma_2^{\text{normal}}, \sigma_3^{\text{normal}})$ is reached.

5.5 | Auditors' reputation

The smart contract *CMSC* can ensure that the auditor selected from the crowd is largely independent. Use a random method to select and exclude related auditors based on various conditions. The incentive function designed can make the auditor has to say the truth to make the most profit. However, a certain mechanism is still needed in *CMSC* to exclude malicious or unreasonable auditors. In our design, all audit interactions are permanently stored on the blockchain. Therefore, we can classify auditors through the audit history stored on the chain, without having to evaluate their reputation from others' feedback.⁴¹

During the audit process, there are often lazy auditors, that is, they only report the audit results randomly or report the results of the last reward without real calculations. This makes it possible

TABLE 1 Strategy and payoff functions for a 3-auditor game

Auditor ₁				σ_1 : Normal	σ_1 : Abnormal
Auditor ₂	σ_2 : Normal	Auditor ₃	σ_3 : Normal	(1, 1, 1)	(−1, 1, 1)
			σ_3 : Abnormal	(1, 1, −1)	(5, −2, 5)
	σ_2 : Abnormal	Auditor ₃	σ_3 : Normal	(1, −1, 1)	(5, 5, −2)
			σ_3 : Abnormal	(−2, 5, 5)	(5, 5, 5)

for lazy nodes to obtain audit rewards without wasting computing resources. For the stored data, it may remain intact most of the time. Therefore, a lazy auditor may report that the data are complete each time without authenticity verification. This leads to our incentive mechanism to continuously reward its behavior. And when the data become abnormally incomplete, although punishments have also occurred, the lazy ones may have already earned incentives before.

To exclude these fraudulent auditors in the process of CMSC's selection of AC, we combine the reputation value of auditors to realize the selection of honest auditors. For example, when the data are confirmed to be complete, the reputation of the auditor who reports the abnormal data decreases by 1; when the data are confirmed to be abnormal, the reputation of the auditor who reports the complete data decreases by 1. When the auditor's reputation value is reduced to 0, the selection of AC members will be automatically blocked by the auditor selection algorithm in the CMSC. Therefore, such lazy auditors avoid appearing in AC. The condition of deterring fraud auditors through reputation is reflected in our Algorithm 1, which is also applied in the specific implementation of smart contracts. The mechanism of blocking fraud auditors through reputation also avoids the unfairness of AC composition caused by the third-party intervention.

The value of the reputation can be set dynamically according to the specific environment. For example, in historical audit tasks, if there are many cheating auditors, the reputation value can be set as small as possible, so that the cheating auditors can be excluded from the audit task as soon as possible. In our paper, if we initialize the reputation value to 10, then every auditor will decrease by 1 when a violation occurs. After accumulating 10 violations, the auditor cannot continue to participate in subsequent audit tasks. All violations are also recorded on the blockchain.

6 | SECURITY ANALYSIS

In this section, we prove that the proposed scheme is secure in terms of correctness, credibility, and the confidence level (CL) of detectability.

6.1 | Correctness

As long As the *Edge* properly stores the *DO*'s file, the proof it generates can pass the final verification of the *ATSC*.

Proof. The proof generated by the *Edge* needs to pass the following three steps to finally decide the correct result.

1. When the Edge properly stores the DO's file, the proof it generates can pass the verification of the auditors in AC.

Given a valid proof $P = \{\mu, \sigma\}$ from the *Edge*, the verification equation (1) in ProofVerify algorithm will hold. On the basis of the properties of Bilinear Map, the verification equation (1) can be proved correct by deducing the left-hand side from the right-hand side:

$$\begin{aligned}
 e(\sigma, g) &= e\left(\prod_{i \in I} \sigma_i^{v_i}, g\right) \\
 &= e\left(\prod_{i \in I} (H(\text{name}||i)^{v_i}) \cdot u^{v_i b_i}, g\right) \\
 &= e\left(\prod_{i \in I} H(\text{name}||i)^{v_i} \cdot u^\mu, g^a\right),
 \end{aligned} \tag{2}$$

where g^a is the public key *pubKey*.

2. The results of auditors in AC can be submitted to ATSC without being plagiarized.

Each auditor in *AC* generates a random secret number $S_k \in Z_p^*$ and utilizes the one-way secrete hash function *Keccak256* to generate the commitment in the first submission stage. The random secret number ensures that other auditors cannot guess the real results submitted.

3. The auditors in *AC* report the audit results truthfully, and the final result in ATSC can correctly reflect the status of the data stored by *Edge*.

We proved in Section 5.4 that the auditors in *AC* must report truthfully based on their audit results to maximize the benefits, otherwise, they will be punished.

6.2 | Credibility

First, in *Crowdauditing*, *DO* needs to send Φ to *Edge*, *Edge* can verify the correctness of Φ . Second, for a challenge initiated by the auditor, *Edge* must compute the proof response to the auditor. Deleting or tampering with data will result in failed generating the proof. On the basis of the cryptographic theory of *Computational Diffie–Hellman Problem* and *Discrete Logarithm Problem*,³⁹ *Edge* cannot or has insufficient computing power to forge proofs. Then, for auditors in *AC*, through our unbiased selection algorithm, we can ensure that the probability of selected *AC* auditors for collusion is quite small, and the Nash equilibrium theory also ensures that *AC* auditors must audit honestly and report truthfully.

6.3 | CL of detectability

We define the *CL* as the probability of successfully detecting an anomaly. Besides, our audit mechanism is mainly used for relatively big files (such as files bigger than 10M), while for smaller files, we can directly challenge a complete file. If the file F is divided into n blocks, and

the probability of block damaged or lost is p , that is, $n \cdot p$ blocks are damaged or lost on average, and a one-time audit can detect nonduplicate random m blocks. The probability that m blocks do not coincide with the damaged blocks is

$$P = \frac{n - n \cdot p}{n} \times \frac{n - n \cdot p - 1}{n - 1} \times \dots \times \frac{n - n \cdot p - m + 1}{n - m + 1}$$

$$= \prod_{i=0}^{m-1} \frac{n - n \cdot p - i}{n - i} \quad (m, n, i \in \mathbb{N}^+, 0 \leq p \leq 1).$$

$$\text{Then, } CL = 1 - P = 1 - \prod_{i=0}^{m-1} \frac{n - n \cdot p - i}{n - i}.$$

Considering that we use very small blocks, such as 10 bytes, and our method is mainly used for relatively big files, such as 10 M, or even larger, our method thus can satisfy that the number of selected blocks m is much smaller than n . In one challenge, $CL = 1 - \prod_{i=0}^{m-1} (1 - p) = 1 - (1 - p)^m$. After k times of challenges, similarly, $CL' = 1 - (1 - CL)^k$.

The CL is only related to the abnormal proportion p and the number of challenging blocks m and has nothing to do with the file size. Figure 5 shows the CL under a different damage ratio p . It can be seen from Figure 5 that when $p = 1\%$ and 10% , higher confidence can be achieved by challenging fewer blocks. Even with a low CL, that is, $p = 0.1\%$, challenging 550 blocks at once can only obtain $CL = 0.423$. But, considering that our challenge is a repeatable process, that is, nine times, the confidence could be increased to $CL' = 1 - (1 - 0.423)^9 = 0.993$, which meets our confidence requirements.

7 | IMPLEMENTATION AND EVALUATION

There are mainly two parts in the implementation and evaluation process: one is the analysis and experiment of the Crowdauditing model, and the other is the smart contract implementation and measurements on the gas consumption of interfaces. Note that “gas” refers to the fee required to successfully conduct a transaction or execute a contract on the Ethereum blockchain platform.

7.1 | Audit performance analysis

We denote n is the total number of data blocks. c is the number of challenged data blocks. $|n|$ is the size of an element of set $[1, n]$, $|p|$ is the size of an element in \mathbb{Z}_p^* , and $|q|$ is the size of an

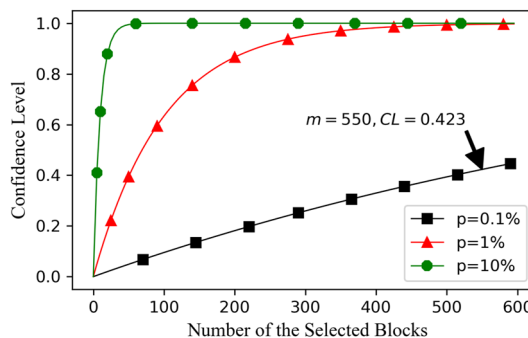


FIGURE 5 Confidence level [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/m.22348)]

element in \mathbb{G}_1 . We analyze the performance of *Crowdauditing* from three aspects: computation, communication, and storage.

Computation: We compare the complexity with two classic schemes, Wang et al.'s scheme⁴² and Huang et al.'s scheme,²³ from the sides of the *DO* and *TPA*. The specific complexity values are shown in Table 2. The *DO* only performs one pairing operation. Compared with traditional methods that use more complex algorithms to defend against antimalicious *TPA*, the design of *Crowdauditing* reduces the computational burden of each auditor in *AC*. This is in line with the idea that the user-owned idle computing resources with limited power consumption constitute the auditor committee in edge storage.

Communication: In the phase of integrity auditing, the auditor in *AC* sends a challenge $chal = \{i, v_i\}_{i \in I}$ to the *Edge*. The size of an auditing challenge is $c \cdot (|n| + |p|)$ bits. Then, the *Edge* generates an auditing proof $P = \{\mu, \sigma\}$ to reply the auditor. The size of an auditing proof $P = \{\mu, \sigma\}$ is $|p| + |q|$ bits. Therefore, for one auditing task, the whole communication overhead is $c \cdot |n| + (c + 1) \cdot |p| + |q|$ bits.

Storage: For each file *Auditors* in *AC* store the data tag τ and the *Edge* store $\Phi = \{\sigma_i\}_1^n$ for the blocks, making the storage overhead to be roughly $(n + 1) \cdot |q|$. During the experiments, we used files containing 500 and 1000 blocks, which gives the storage overhead of 11 and 21 KB, respectively.

Comparison: Considering the length of the paper, we mainly show the superiority of decentralized audit compared with a single *TPA* audit. We assume that a set of tasks consume the same computing power and take the task as the unit to compare the number of tasks processed by each auditor in *AC* and the traditional audit with a single *TPA*. Besides, we assume that there are 10,000 tasks and 1000 auditors in the edge storage system.

From Figure 6A, we can see that the number of audit tasks undertaken by auditors in *Crowdauditing* is much smaller than the traditional audit model. Besides, when the number of auditors in an *AC* increases, the number of *AC* groups decreases, and the number of tasks each auditor undertakes also increases, which can effectively improve the reliability of the results. This situation is more pronounced in Figure 6B. This is also in line with the concept of decentralized auditing using participates with limited resources in edge scenarios.

7.2 | Smart contract implementation and analysis

According to the *Crowdauditing* model, we implement a prototype system based on Ethereum smart contract.

7.2.1 | CMSC implementation

As shown in Figure 7, we divide the Auditor state into four types: "Online," "Offline," "Await," and "Busy." The judgment box represents the choice in case of conflict in the contract. We use the form of "Role: Interface Name" to indicate the ownership and name of the interface. When the auditor registers to join the crowd, the auditor has one state, "Online" or "Offline," and the auditor can switch between the two states according to its own situation. Committee members can only be selected from nodes whose status is "Online." When using Algorithm 1 to select Committee members from the crowd one by one, the status of the auditor changes from

TABLE 2 Theoretical performance comparison with one challenge

Mechanism	DO's overhead	TPA's overhead	Antimalicious TPA	Followability	Concurrency
Wang et al. ⁴²	Small	$(2 + c)M + (1 + c)E + 2P$	No	No	Low
Huang et al. ^{2,3}	$2cM + cE + 1TEnc$	$(2 + c)M + (3 + c)E + 2P + 1TDec$	Yes	No	Low
Ours	$1P$	$cM + (1 + c)E + P$	Yes	Yes	High

Notes: M denotes one multiplication operation, E denotes one exponentiation operation, P denotes one pairing operation, Div denotes one division operation, $TEnc$ is the timed-release encryption,⁴³ $TDec$ is the timed-release decryption,⁴³ and c is the number of sampled blocks.

Abbreviations: DO, Data Owner; TPA, third-party auditor.

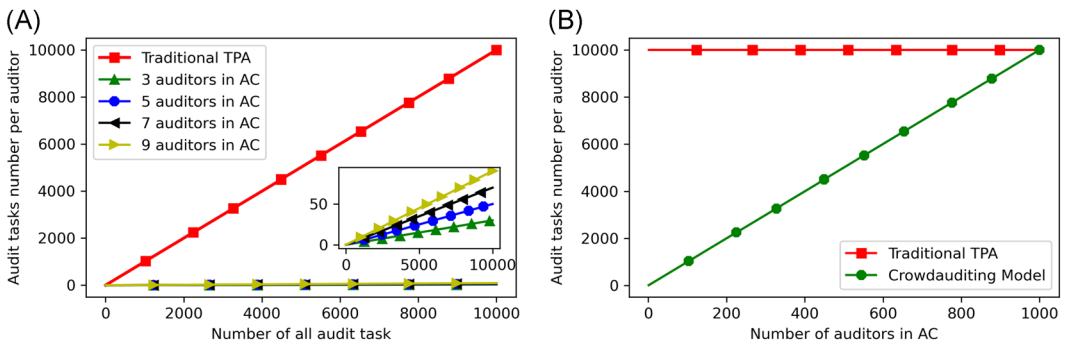


FIGURE 6 (A) Task number per auditor when different AC auditor numbers. (B) Task number per auditor when different AC auditor numbers [Color figure can be viewed at wileyonlinelibrary.com]

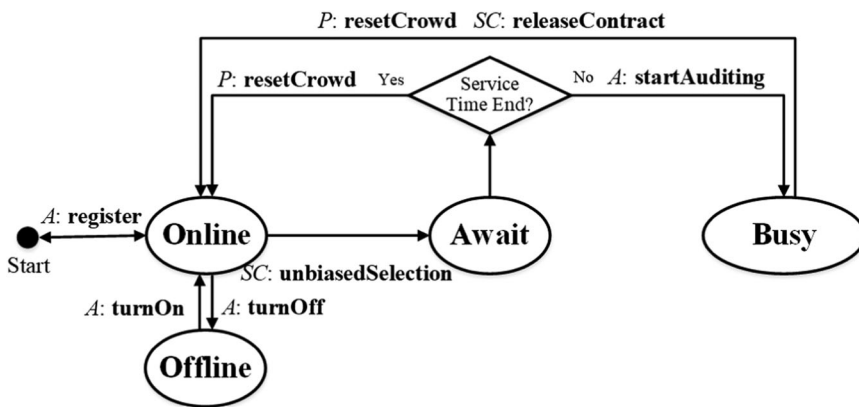


FIGURE 7 Crowd smart contract implementation

“Online” to “Await.” After the committee members are selected, the audit task is started, and the status of the auditor changes from “Await” to “Busy.”

7.2.2 | ATSC implementation

Figure 8 shows the contract state transition when implementing our contract. The ATSC involves three roles: Client (C), Auditor (A), and Service Provider (P). The transition states in the contract are “Init,” “Active,” “Commitments,” “Reveal,” and “Complete.” The interfaces designed for ATSC are named in the form of “ $R_{\text{role}}:N_{\text{interface}}$.” It means that only the role R_{role} can invoke interface $N_{\text{interface}}$. In different states, the checking mechanism is the property of the programming language provided by Ethereum, specific roles can access specific interfaces, to realize the interaction between roles and the contract.

In the blockchain, the transition of the contract from one state to another needs to be realized by triggering to call functions, and it also needs to pay a certain cost. For example, in Figure 8, when the current round of the audit task is completed, the provider can restart the audit task by calling interface “ $P: \text{restartAudit}$ ” or “ $P: \text{resetAudit}$.” When the interface call ends, the state of the ATSC is converted from “Complete” to “Init” or “Active.”

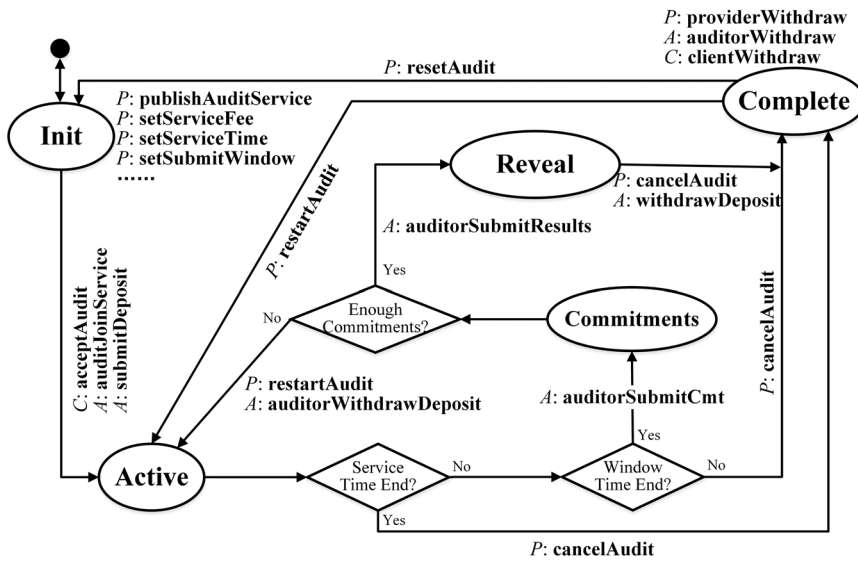


FIGURE 8 Audit task smart contract implementation

7.2.3 | “Gas” consumption of each interface in smart contracts

We adopt Online Ethereum Studio to develop the Ethereum smart contracts, which supports simulation for the contract automatic deployment, multiuser establishment, and users can interact with contracts through a web client. Users can also test the “Gas” consumption of interfaces in the contract, which is the transaction fee and costs Ethereum cryptocurrency “Ether.” To simulate the real application scenario, we test all interfaces of the *Crowdauditing* by deploying a contract on the Ethereum blockchain test network “Rinkeby.”¹⁵ Rinkeby is a global blockchain test network for developers to debug smart contracts. It can simulate the secret currency “Ether” in the real Ethereum blockchain, which is convenient for us to debug the interface. We generate multiple accounts on “Rinkeby” to simulate different roles and take the “Ether” retrieved from each simulated account to execute the interface and prepay different types of fees. The provider generates an ATSC according to the audit details negotiated with the client off-chain. We then test all possible scenarios to leverage and validate the functionality of the different interfaces.

On the basis of a carefully designed decentralized integrity audit scheme, this paper proposes a two-stage result submission strategy for the AC, which avoids the possibility of off-chain collusion among auditors or replication of other’s results. Moreover, the credibility of the results submitted through the two-stage submission strategy is ensured by payoff function design and further proved through game theory. Therefore, we mainly analyze some performances from experimental research. The performance of the smart contract is mainly reflected by the interfaces. The complexity of the interface determines the cost of state transition. In the Ethereum network, miners need to consume electric energy for executing the program defined in the interface to verify, so a higher complexity of the interface implies a higher cost. The “Gas” defined in Ethereum is used to measure the execution cost. It is a unit used to represent the work done by miners in executing transactions. Transaction cost is the product of gas consumption and price per unit of natural

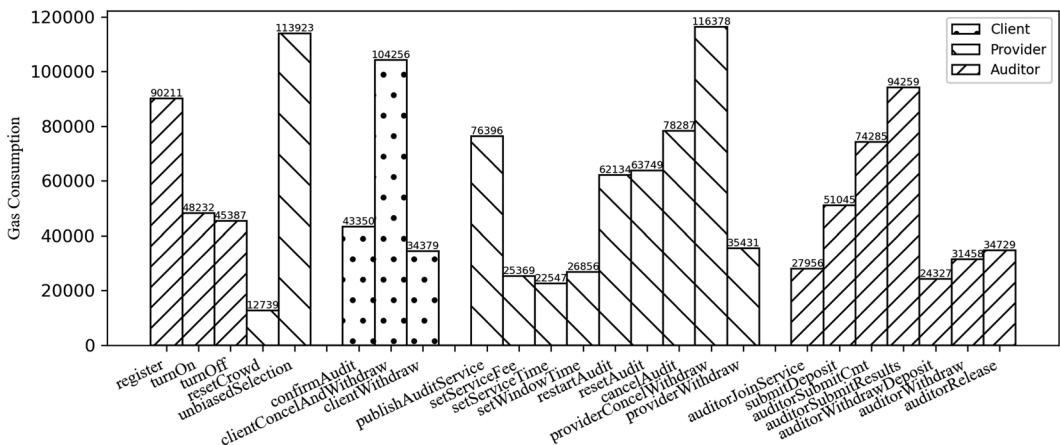


FIGURE 9 The Gas consumption of each interface in ATSC

gas. Therefore, the gas consumption is similar in either the test network or the main network. We recorded all gas consumption for each interface from the transaction history of the experiment.

Figure 9 displays the gas consumption of the main interface in the contract. It reveals that the interfaces of *SP* consume more gas. Since the working capacity of the Client's equipment is different, some work is transferred to the provider, which meets the requirements of Clients in practice. Auditors also consume less gas, which is also in line with the application situation in real scenarios.

8 | CONCLUSIONS

To the best of our knowledge, this is the first work developing a practical *Crowdauditing* model using blockchain for decentralized data integrity in the edge storage service. We use a *Crowdauditing* method to realize the decentralized data integrity audit and propose a two-stage submission strategy to strengthen the trustworthiness of off-chain results. Specifically, we design a payoff function for the auditor committee and leverage game theory to incentivize the untrusted auditors to provide honest audit services for maximizing their revenue. In this way, the issue of trust is transformed into an economic issue, that is, economic principles force the auditors to report truthful audit results. The credibility of the auditor is not essential anymore, and the auditing task can be delivered to the decentralized crowd for improving the auditing quality and reliability. The limitation of the system is that there must be enough auditors to participate, so that malicious users cannot achieve the purpose of controlling the results by registering a large number of false auditors. Finally, we fully implemented *Crowdauditing* using Ethereum smart contracts and performed an experimental study to demonstrate its feasibility and system performance. We demonstrated the gas consumption of the interface in the prototype system in the final experiment. Some interfaces have higher consumption values. In future work, we plan to continue to optimize the interfaces to reduce consumption. In addition, we are also further studying the scenarios where the *Crowdauditing* intelligent audit model can be applied to other scenarios in more fields.

ACKNOWLEDGMENTS

This study is supported by the National Natural Science Foundation of China (62072465), the National Key Research and Development Program of China (2018YFB0204301 and 2020YFC2003400), and the NUDT Research Grant (No. ZK19-38).

ENDNOTES

*The smart contract we implemented: <https://github.com/haiwen1128/SmartContracts4Crowdauditing>

ORCID

Haiwen Chen  <https://orcid.org/0000-0001-8031-8008>

Huan Zhou  <https://orcid.org/0000-0003-2319-4103>

Jiaping Yu  <https://orcid.org/0000-0003-4367-3179>

Kui Wu  <https://orcid.org/0000-0002-2069-0032>

Fang Liu  <https://orcid.org/0000-0001-8753-3878>

Tongqing Zhou  <https://orcid.org/0000-0002-6620-1898>

Zhiping Cai  <https://orcid.org/0000-0001-5726-833X>

REFERENCES

- Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Internet Things J.* 2016; 3(5):637-646.
- Micron Edge Storage: Optimizing Storage Costs for Video Surveillance Systems. <https://www.micron.com/solutions/video-surveillance>
- Yu J, Chen H, Wu K, Zhou T, Cai Z, Liu F. Centipede: leveraging the distributed camera crowd for cooperative video data storage. *IEEE Internet Things J.* 2021;1:1-13. <https://doi.org/10.1109/JIOT.2021.3074823>
- Chen H, Yu J, Liu F, Cai Z, Xia J. Archipelago: a medical distributed storage system for interconnected health. *IEEE Internet Comput.* 2019;24(2):28-38.
- Secure Enterprise File Services from Edge to Cloud. <https://www.ctera.com>
- Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores. In: *Proceedings of the 14th ACM Conference on Computer and Communications Security, October 2007, New York, NY, USA*; 2007. <https://doi.org/10.1145/1315245.1315318>
- Zheng Z, Xie S, Dai H, Chen X, Wang H. An overview of blockchain technology: architecture, consensus, and future trends. In: *2017 IEEE International Congress on Big Data (BigData Congress), June 25-30, 2017, Honolulu, HI, USA*; 2017. <https://doi.org/10.1109/BigDataCongress.2017.85>
- Liu A, Du X, Wang N, Li S. Research progress of blockchain technology and its application in information security. *J Softw.* 2018;29(7):2092-2115.
- Buterin V. A next-generation smart contract and decentralized application platform. *White Paper.* 2014; 3(37). https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf
- Osborne MJ. *An Introduction to Game Theory.* Toronto, Canada: CRC Press; 2004.
- Myerson RB. *Game Theory.* Cambridge, MA, USA: Harvard University Press; 2013.
- Wu H, Zhang J, Cai Z, et al. Resolving multi-task competition for constrained resources in dispersed computing: a bilateral matching game. *IEEE Internet Things J.* 2021;1:1-12. <https://doi.org/10.1109/JIOT.2021.3075673>
- Chen H, Yu J, Zhou H, Zhou T, Liu F, Cai Z. SmartStore: a blockchain and clustering based intelligent edge storage system with fairness and resilience. *Int J Intell Syst.* 2021;1:1-26. <https://doi.org/10.1002/int.22509>
- Holt CA, Roth AE. The Nash equilibrium: a perspective. *Proc Natl Acad Sci.* 2004;101(12):3999-4002.
- Rinkeby: Ethereum Testnet. <https://www.rinkeby.io/#stats2021>
- Bowers KD, Juels A, Oprea A. Proofs of retrievability: theory and implementation. In: *Proceedings of the 2009 ACM Workshop on Cloud Computing Security, New York, NY, USA*; 2009. <https://doi.org/10.1145/1655008.1655015>

17. Deswarte Y, Quisquater J, Saïdane A. Remote integrity checking. In: *Working Conference on Integrity and Internal Control in Information Systems*, Boston, MA; 2003. https://doi.org/10.1007/1-4020-7901-X_1
18. Juels A, Kaliski BS Jr. PORs: proofs of retrievability for large files. In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*, October 2007, New York, NY, USA; 2007. <https://doi.org/10.1145/1315245.1315317>
19. Hao Z, Zhong S, Yu N. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE Trans Knowl Data Eng*. 2011;23(9):1432-1437.
20. Liu H, Zhang P, Liu J. Public data integrity verification for secure cloud storage. *J Netw*. 2013;8(2):373.
21. Li J, Tan X, Chen X, Wong DS, Xhafa F. OPoR: enabling proof of retrievability in cloud computing with resource-constrained devices. *IEEE Trans Cloud Comput*. 2014;3(2):195-205.
22. Singh AP, Pasupuleti SK. Optimized public auditing and data dynamics for data storage security in cloud computing. *Procedia Comput Sci*. 2016;93:751-759.
23. Huang K, Xian M, Fu S, Liu J. Securing the cloud storage audit service: defending against frame and collude attacks of third party auditor. *IET Commun*. 2014;8(12):2106-2113.
24. Worku SG, Xu C, Zhao J, He X. Secure and efficient privacy-preserving public auditing scheme for cloud storage. *Comput Electr Eng*. 2014;40(5):1703-1713.
25. Xu C, He X, Daniel A. Cryptanalysis of auditing protocol proposed by Wang et al. for data storage security in cloud computing. In: *3rd International Conference on Information Computing and Applications (ICICA)*, September 14-16, 2012, Chengde, China; 2012. <https://eprint.iacr.org/2012/115.pdf>
26. Liu H, Chen L, Davar Z, Pour MR. Insecurity of an efficient privacy-preserving public auditing scheme for cloud data storage. *J Universal Comput Sci*. 2015;21(3):473-482.
27. Liu B, Yu X, Chen S, Xu X, Zhu L. Blockchain based data integrity service framework for IoT data. In: *2017 IEEE International Conference on Web Services (ICWS)*, June 25-30, 2017, Honolulu, HI, USA; 2017. <https://doi.org/10.1109/ICWS.2017.54>
28. Yue D, Li R, Zhang Y, Tian W, Peng C. Blockchain based data integrity verification in P2P cloud storage. In: *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, December 11-13, 2018, Singapore; 2018. <https://doi.org/10.1109/ICPADS.2018.8644863>
29. Hao K, Xin J, Wang Z, Wang G. Outsourced data integrity verification based on blockchain in untrusted environment. *World Wide Web*. 2020;23(4):2215-2238.
30. Zhou H, Ouyang X, Ren Z, Su J, Laat C, Zhao Z. A blockchain based witness model for trustworthy cloud service level agreement enforcement. In: *IEEE INFOCOM 2019—IEEE Conference on Computer Communications*, April 29-May 2, 2019, Paris, France; 2019. <https://doi.org/10.1109/INFOCOM.2019.8737580>
31. Miao Y, Huang Q, Xiao M, Li H. Decentralized and privacy-preserving public auditing for cloud storage based on blockchain. *IEEE Access*. 2020;8:139813-139826.
32. Du Y, Duan H, Zhou A, Wang C, Au MH, Wang Q. Towards privacy-assured and lightweight on-chain auditing of decentralized storage. In: *2020 IEEE 40th International Conference on Distributed Computing Systems*, November 29-December 1, 2020, Singapore; 2020. <https://doi.org/10.1109/ICDCS47774.2020.00023>
33. Wang T, Bhuiyan MZA, Wang G, Qi L, Wu J, Hayajneh T. Preserving balance between privacy and data integrity in edge-assisted Internet of Things. *IEEE Internet Things J*. 2019;7(4):2679-2689.
34. Tong W, Jiang B, Xu F, Li Q, Zhong S. Privacy-preserving data integrity verification in mobile edge computing. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, July 7-10, 2019, Dallas, TX, USA; 2019. <https://doi.org/10.1109/ICDCS.2019.00104>
35. Liu D, Shen J, Vijayakumar P, Wang A, Zhou T. Efficient data integrity auditing with corrupted data recovery for edge computing in enterprise multimedia security. *Multimedia Tools Appl*. 2020;79:10851-10870.
36. Li B, He Q, Chen F, Jin H, Xiang Y, Yang Y. Auditing cache data integrity in the edge computing environment. *IEEE Trans Parallel Distrib Syst*. 2021;32(5):1210-1223.
37. Yue D, Li R, Zhang Y, Tian W, Huang Y. Blockchain-based verification framework for data integrity in edge-cloud storage. *J Parallel Distrib Computing*. 2020;146:1-14.
38. Labovitz C, Ahuja A, Jahanian F. Experimental study of internet stability and backbone failures. In: *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No. 99CB36352)*, June 15-18, 1999, Madison, WI, USA; 1999. <https://doi.org/10.1109/FTCS.1999.781062>

39. Shen W, Qin J, Yu J, Hao R, Hu J. Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. *IEEE Trans Inf Forensics Secur.* 2018;14(2):331-346.
40. Nash JF. Equilibrium points in n-person games. *Proc Natl Acad Sci.* 1950;36(1):48-49.
41. Wang X, Su J, Hu X, Wu C, Zhou H. Trust model for cloud systems with self variance evaluation. In: Nepal S, Pathan M, eds. *Security, Privacy and Trust in Cloud Systems*. Berlin, Heidelberg: Springer; 2013. https://doi.org/10.1007/978-3-642-38586-5_10
42. Wang C, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing. In: *2010 Proceedings of the IEEE INFOCOM, March 14-19, 2010, San Diego, CA, USA*; 2010. <https://doi.org/10.1109/INFOCOM.2010.5462173>
43. Xu J. Auditing the auditor: secure delegation of auditing operation over cloud storage. *IACR Cryptol ePrint Arch.* 2011;2011:304. <https://eprint.iacr.org/2011/304.pdf>

How to cite this article: Chen H, Zhou H, Yu J, et al. Trusted audit with untrusted auditors: A decentralized data integrity Crowdauditing approach based on blockchain. *Int J Intell Syst.* 2021;36:6213-6239. <https://doi.org/10.1002/int.22548>