Hindawi
Security and Communication Networks
Volume 2021, Article ID 6662135, 11 pages
https://doi.org/10.1155/2021/6662135

Research Article

Audit Outsourced Data in Internet of Things

Gaopan Hou,[1,2] **Jianfeng Ma**,[1,2] **Jiayi Li**,[1,2] **and Chen Liang**[3]

[1]*School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[2]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China*
[3]*PLA 94860 Troops, Nanjing 210000, China*

Correspondence should be addressed to Jianfeng Ma; jfma@mail.xidian.edu.cn

Received 20 November 2020; Revised 25 February 2021; Accepted 24 April 2021; Published 8 May 2021

Academic Editor: Shehzad Ashraf Chaudhry

With the increase in network transmission rates, the Internet of Things (IoT) has gradually become a trend. Users can upload the data generated by the device to the cloud database to save local storage space, thereby reducing local storage costs. Because uploading data to the cloud loses physical control of the data, an audit is required. Traditional audit protocols are not completely suitable for lightweight devices in the IoT. This paper proposes a new type of audit protocol suitable for lightweight devices with weak computing power. This protocol transfers part of the computation of data tags to a cloud storage provider (CSP) with strong computing power, thereby reducing the introduction of computing entities. Our scheme supports the dynamic operation of data and guarantees the effectiveness of challenge response by blockchain. Compared with existing schemes, our scheme is more secure and effective.

1. Introduction

Due to the large-scale application of 5G, the Internet of Things has developed rapidly. At the same time, many emerging technologies have emerged, such as cloud storage [1, 2]. Because of a series of advantages such as the scalability and lower upfront cost of cloud storage, more and more entities choose to store data in the cloud. With cloud storage, users are free from the physical limitations of local devices and can store and share their data anytime, anywhere. So far, many studies have focused on the cloud [3–6]. Gartner's latest cloud computing market tracking data shows that the global cloud computing infrastructure service market continued to grow rapidly in 2019, with a year-on-year growth of 37.3% [7].

Although cloud storage brings great convenience, risks follow. The main risk of cloud storage is that users upload their data to the cloud server and lose physical control of the data. At the same time, the cloud service provider is not completely reliable. They may suffer downtime or attack and passively lose users' data. To make matters worse, CSP may actively discard users' infrequently accessed data to reduce its own operating costs. Frequent cloud security incidents have aggravated people's concerns about cloud security and hindered the development of cloud storage [8].

In order to ensure the integrity of data, promptly detect dishonest behaviors of CSP, and urge CSP to provide high-quality storage services, cloud security audits have gradually become a hot issue for cloud storage. Through auditing, users can know whether the data stored in the cloud has been damaged and at the same time can effectively supervise the services provided by the CSP.

Traditional cryptographic schemes cannot be directly used for auditing. It is impractical to download the data in cloud storage directly for verification because of the excessive overhead. With the deepening of research, proof of retrievability (PoR) and provable data possession (PDP) models have been proposed one after another. The data owner sets a small data tag for each data block. During inspection, by concentrating the selected data block and its tag in a small piece of evidence, it can be detected with a high probability whether the data block has suffered damage. In order to reduce the burden on users, a trusted third party is introduced to perform audits instead of users, thus realizing public auditing.

Existing public audit schemes have been able to support the audit of dynamically updated data, by constructing hash tables, Merkle hash tree, and other data structures and at the same time with the help of auxiliary authentication information. Although good progress has been made, there are still many problems. We have conducted specific research and found the following deficiencies. First of all, in the data preprocessing stage, users need to calculate a large number of data block tags on their own, which is not friendly to devices with weak computing power. Secondly, TPA needs to store some additional information. In the case of multiple users, the storage capacity will increase. The current public audit relies on the assumption that the TPA is honest. Once the TPA colludes with the CSP, the entire public audit becomes invalid. In addition, if the user wants to limit the dishonest behavior of TPA by checking the audit log of TPA, the entire audit protocol will still be invalid due to the existence of collusion attacks, because TPA and CSP can manually select intact data blocks for auditing, which is not detected on the log.

After reviewing the shortcomings of previous schemes, on the basis of our previous research [9], we proposed an efficient outsourcing audit scheme for lightweight devices in the IoT environment. Our contributions are as follows:

(1) We have conducted research on the outsourcing audit of lightweight devices in the IoT environment. We have designed an audit protocol suitable for lightweight devices. And it proved the correctness of the proposed scheme under the PDP model.

(2) Different from the practice of outsourcing computing tasks to TPA in most schemes, considering that the final data is stored in CSP, CSP has powerful computing capabilities, so we directly outsource computing tasks to CSP to avoid waste of resources.

(3) Different from the TPA selected challenge block adopted by most schemes, we generate challenge blocks based on the Ethereum network, effectively avoiding collusion attacks.

2. Related Work

In recent years, with the increasing maturity of cloud storage technology, more and more users choose to outsource data to the cloud. Therefore, cloud security auditing has become particularly important. Blum et al. [10] conducted research on cloud auditing for the first time. Juels and Kaliski proposed proof of retrieval (PoR) [11], which is mainly used for static archiving of large files. Shacham and Waters [12] designed an improved PoR scheme for [11]. They used a publicly verifiable homomorphic verifier constructed by BLS signatures [13] to concentrate the proofs into small tags. Unfortunately, it still does not support dynamic operations. Ateniese et al. [14] defined a provable data possession (PDP) model based on a homomorphic linear authenticator constructed with RSA. This is a probabilistic verification model that allows auditors to achieve integrity verification without retrieving the entire file from the cloud server. Their follow-up work [15] still does not support fully dynamic operations. Erway et al. [16] extended the PDP model and proposed a model for dynamic provable data possession (DPDP), which uses a level-based authentication skip list to perform provable updates to stored data. Nevertheless, computation and communication costs are still very expensive. Wang et al. [17] considered dynamic data storage in a distributed situation, and the challenge-response protocol can both determine the correctness of the data and find possible errors. Similar to [15], they only consider partial support for dynamic operations. Later, they proposed an audit scheme that supports privacy protection [18]. Then, in [19], they improved the previous PDP model by manipulating the Merkle hash tree for block tag authentication. Zhu et al. [20] discussed multicloud storage and proposed a collaborative PDP scheme that can effectively support data migration. Yang and Jia [21] proposed an audit scheme that supports dynamic attributes and privacy protection attributes. Armknecht et al. [22] proposed a privately auditable PoR scheme that can be delegated audit while preventing malicious clients, auditors, and cloud servers from colluding attacks. Liu et al. [23] pointed out that MHT itself is not sufficient to verify the block index, which may lead to replacement attacks. They provide a top-down multireplica MHT data audit scheme for dynamic big data storage in the cloud. Guan et al. [24] proposed the first cloud storage audit scheme based on indistinguishable confusion. Wang et al. [25] designed a novel identity-based proxy-oriented remote data integrity auditing scheme by introducing a proxy to process data for users. Sookhak et al. [26] presented a new data structure named divide and conquer table, which can be used for the auditing of big data storage. In [27], a Merkle hash tree based trusted third-party auditing of big data was proposed. Shen et al. [28] proposed a remote data integrity auditing scheme based on sensitive information hiding. In their scheme, sensitive information is protected while other information is not affected. Rao et al. [29] presented a new approach, based on batch-leaves-authenticated Merkle hash tree, to batch-verify multiple leaf nodes and their own indexes altogether.

3. Preliminaries

3.1. Bilinear Pairing. Bilinear pairing [30] refers to the corresponding linear mapping relationship between two cyclic groups. Since the set formed by all points on the elliptic curve will form a group relationship in algebraic geometry, the operation of the bilinear pairing function can be applied to the elliptic curve.

Define three multiplicative cyclic groups G_1, G_2, and G_3 in Z_q, define the mapping e: $G_1 \times G_2 \longrightarrow G_3$, satisfying the following:

(1) Bilinear: for $\forall g_1 \in G_1, g_2 \in G_2, \forall a, b \in Z_q$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds

(2) Nondegeneration: $\exists g_1 \in G_1, g_2 \in G_2$, s.t. $e(g_1, g_2) \neq 1_{G_3}$

(3) Computability: efficient algorithms exist for $\forall g_1 \in G_1, g_2 \in G_2$, $e(g_1, g_2)$ can be calculated

3.2. Discrete Logarithm Assumption. Diffie and Hellman introduced the concept of public key cryptography [31] in 1976. Since then, the security of most cryptosystems depends on the discrete logarithm assumption. Odlyzko [32] gave a detailed explanation of this. Once the assumptions become easy to solve, the corresponding systems will be broken. The security of the scheme depends on the following difficult problem:

CDH problem: computational Diffie–Hellman problem is that, given $g, g^x, g^y \in G$, it is difficult to calculate g^{xy} for unknown $x, y \in Z_q^*$

DLP problem: discrete logarithm problem is that, given $v \in G$, solving for integers $\alpha \in Z_q^*$, s.t. $v = g^\alpha$ established

CDHI problem: computational Diffie–Hellman inverse problem is that, given $g^x \in G$ ($x \in Z_q^*$ unknown), output $g^{x^{-1}}$

3.3. BLS Homomorphic Signature. BLS signature [13] is a short signature scheme constructed by Boneh et al. using bilinear pairing. Due to the short signature length under the same security strength, it has a huge advantage in communication and storage overhead.

BLS signature scheme consists of three algorithms, where $H(\cdot)$: $\{0,1\}^* \longrightarrow G$ is the anticollision hash function:

(1) KeyGen: the algorithm selects a random element α from the finite field Z_q and then calculates $v \longleftarrow g^\alpha$ as a public key. A private and public key pair (α, v) can be obtained.

(2) Sign: given message m, calculate $h = H(m)$, and then sign it with α to get $\sigma = h^\alpha$.

(3) Sigverify: given v, m, and σ, calculate $H(m)$, and then use the properties of bilinear mapping to verify whether $e(\sigma, g) = e(H(m), v)$ is established. If satisfied, then σ is the BLS signature of message m, and message m is complete.

3.4. Merkle Hash Tree with Relative Nodes. Merkle hash tree was proposed by Merkle [33]. It is essentially a tree-like data structure, which is widely used in data integrity verification. It is a process of constantly repeating the hash and finally forming a root hash. If there is a way to confirm the correctness of the root hash, then to confirm the integrity of a certain data block of the file only needs to complete the hash calculation process from several leaf nodes up to the root node of the tree. In this paper, in order to ensure the correct location of the data block and prevent data block replacement attacks, we have introduced auxiliary authentication information into the hash tree to achieve the binding of the data block and the location information: Let Num be the number of nodes from the root node to the target node, that is, $\text{Num}_{\text{root}} = 1$. In order to locate the node direction, we introduce Dir to represent the node direction. If Dir = 0, it means the node belongs to the left child node. If Dir = 1, it means the node belongs to the right child node. At the same time, we stipulate that Dir = 2 at the root node.

3.5. Blockchain. Blockchain [34] is a new application mode of computer technology such as distributed data storage, point-to-point transmission, consensus mechanism, and encryption algorithm. It is a series of data blocks associated with cryptographic methods. It is essentially a decentralized database. Each data block contains a batch of Bitcoin network transaction information to verify the validity of the information. A block is a storage unit, which records all the communication information of each block node within a certain period of time. Each block is linked by random hashing. The next block contains the hash value of the previous block. With the expansion of information exchange, a block and a block are successively connected, and the result is called a blockchain. Due to the characteristic of decentralization, it has received more and more extensive research [35].

4. Problem Statement

The local storage space of IoT devices is limited, and the successively generated data may exceed the storage space of the device itself. Therefore, the data generated by the device can be uploaded to the CSP through the network to reduce the cost of local storage hardware. Although the existing audit protocol has achieved rapid development, there are still a lot of problems. The first is the problem of computing overhead. IoT devices do not need very powerful computing power to complete the calculation of data tags. The second problem is the effectiveness of auditing, and the invalidation of audit results caused by collusion attacks should be fully avoided. Finally, there is the issue of the timeliness of the audit. It should be ensured that the third party can perform the audit in a timely manner and record the results. Therefore, in response to the above problems, we have designed a more efficient and safe audit scheme.

5. System Model

In our system model, there are 3 different participants, namely, user, CSP, and TPA. The characteristics of the 3 participants are as follows:

User: owner of the local device of IoT who uploads the data to CSP and entrusts maintenance, management, and calculation to CSP, meanwhile entrusting data audit service to TPA

CSP: composed of many cloud servers, with strong computing and storage resources

TPA: help users to audit data stored on cloud servers

6. The Proposed Scheme

6.1. Notations and Definitions. Table 1 shows all the notations and their corresponding definitions used in this paper.

6.2. Details. In this section, we will describe the specific content of the scheme. In most existing schemes, the user uploads a series of information (such as the data block with its tags) to CSP after all the data is preprocessed locally. For

Table 1: Notations and definitions.

Notation	Definition
λ	Safety strength parameters
Z_q	q-order finite field
G	q-order multiplicative cyclic group
g	Generator of G
α_u	User's private key
v	User's public key
D	Data outsourced to the cloud
d_i	User's i-th data block
σ_i	Homomorphic tag of d_i
Ξ	Auxiliary calculation information generated by CSP
Φ	Homomorphic tag collection
e	Bilinear mapping
n	Number of data blocks
s	Number of data blocks challenged
r_i	Random number of the challenge block pairing
I_i	Index of data block d_i
Ω_i	Auxiliary authentication information of nodes
P	Challenge-response proof generated by CSP
$H(\cdot)$: $\{0,1\}^* \longrightarrow G$	Collision-resistant one-way hash function
H_R	Root hash

the IoT environment, considering the storage and computation overhead of the local device, we upload the encrypted data to CSP firstly. Then, we perform a series of calculations such as data tag and MHT generation with the assistance of CSP, which greatly reduces the overhead of devices. The specific details are as follows.

6.2.1. Data Outsourcing Stage. Before data uploading, global parameter generation, key generation, and data split are required. After successful uploading, DataPreproc, TagGen, and TagVerifywill be executed next, specifically:

GlobalPmGen: user selects a parameter λ for the IoT devices according to the required security strength s.t. the large prime number q satisfies $\log_2 q \le \lambda$, selects the multiplicative cyclic group G whose generator is g in Z_q, and selects the collision-resistant one-way hash function $H(\cdot)$: $\{0,1\}^* \longrightarrow G$.

KeyGen: user selects α_u in Z_q as the private key for signature and calculates $v = g^{\alpha_u}$ as the corresponding public key.

DataSplit: user divides the data into n blocks, namely, $D = (d_1, d_2, \ldots, d_n)$, $i = 1, 2, \ldots, n$, and then, uploads them to CSP.

DataPreproc: after receiving d_i, CSP generates a series of auxiliary information Ξ for the user, including $H(d_i)$, v^{d_i} and MHT. Specifically, CSP uses $H(\cdot)$ to hash all d_i to obtain $H(d_i)$, then constructs MHT, and calculates v^{d_i} according to the user's public key v. After completion of the calculation, $\Xi - \{\text{MHT}, v^{d_i}\}$ is sent to the user.

TagGen: user randomly selects 460 hash values in the returned MHT for verification. If the verification fails, CSP will be warned and required to recalculate. If it is passed, $H(d_i)^{\alpha_u}$ will be calculated, and then the tag σ_i, $i = 1, 2, \ldots, n$ will be generated by (1) according to Ξ. Therefore, the user can get the homomorphic tag set $\Phi = \{\sigma_i \mid i = 1, 2, \ldots, n\}$. Then, the user signs the root hash H_R: $\sigma_{H_R} = (H_R)^{\alpha_u}$ with α_u. Finally, the user sends $\{\Phi, \sigma_{H_R}\}$ to CSP:

$$\sigma_i = H(d_i)^{\alpha_u} \cdot v^{d_i}. \tag{1}$$

TagVerify: after receiving $\{\Phi, \sigma_{H_R}\}$, CSP verifies the correctness of σ_{H_R} by

$$e\left(\sigma_{H_R}, g\right) = e\left(H_R, v\right). \tag{2}$$

After passing (2), CSP verifies the correctness of each homomorphic tag by

$$e(\sigma_i, g) = e\left(H(d_i) \cdot g^{d_i}, v\right). \tag{3}$$

If (3) fails, CSP refuses to accept and then asks the user to resend it; if it holds, CSP returns the proof of storage to the user. After the user receives it from CSP, they delete the local data and only keep the key information.

6.2.2. TPA Audit Stage. In order to reduce the burden on users, it is a common choice to delegate audit tasks to TPA. Here, TPA can be any authorized entity. Therefore, the public audit was completed. The TPA audit stage includes the following three algorithms.

ChallengeGen: the system automatically triggers the audit through the Ethereum smart contract. Specifically, if the last ℓ digit ($\ell \in N$) of the current block hash meets the set conditions, the smart contract performs calculations and sequentially obtains the challenge data block number with its corresponding coefficient, where ℓ defines the TPA challenge request period, and at the same time, to ensure the random selection of the challenge block, we generate the challenge request data block tag based on the current hash value and the user ID number. Assuming that s data block indexes $\{I_i\}$ and their corresponding coefficients $r_i \in Z_q$ are selected finally, then, they will be combined into a challenge request Cl: $\{(I_i, r_i)\}_{1 \le i \le s}$ and sent to CSP.

ProofGen: according to Cl, CSP queries the user's data and corresponding tag stored on the server and then generates the integrity storage proof P, which is composed of $\{\mathfrak{R}, \sigma, \text{MHT}, \sigma_{H_R}\}$, where

$$\mathfrak{R} = \sum_{(I_i, r_i) \in Cl} r_i \cdot d_i, \tag{4}$$

$$\sigma = \prod_{(I_i, r_i) \in Cl} \sigma_i^{r_i}. \tag{5}$$

Finally, P is sent to TPA.

ProofCheck: after receiving the proof, TPA reconstructs MHT, then selects 460 hash values from MHT to verify the correctness, and finally verifies the correctness of σ_{H_R} by (2).

If (2) does not hold, it indicates that the data has been corrupted and returns to "refuse"; if (2) holds, the TPA continues to use (6) for judgment. If (6) holds, it returns to "accept," indicating that the data is stored completely:

$$e(\sigma, g) \stackrel{?}{=} e\left(\prod_{(I_i, r_i)\in Cl} \left(H(d_i)^{r_i} \cdot g^{\Re}\right), v\right). \tag{6}$$

After verification, TPA sends 0eth to the user's Ethereum contract account and notes the block number of the challenge response with the audit result.

6.2.3. User Audit Stage. In order to check whether TPA has completed the audit task honestly, the user needs to check TPA's audit logs within a data update cycle (considering the user's cost, this cycle should be much larger than the TPA irregular audit). The user checks the historical transaction information from the Ethereum account and extracts x pieces of challenge information with their corresponding random numbers. Firstly, the user randomly extracts the challenge data block information and calculates whether the selected challenge data block matches the data block randomly generated in the blockchain according to the predetermined algorithm. If not, the TPA is fraudulent. If the detection result is that the TPA honestly selected the random block, then check whether the equation is true through the bilinear pairing operation. If it is true, TPA has completed the user's audit task honestly. If it fails, TPA is fraudulent.

6.2.4. Dynamic Update Stage

(1) Data Block Insertion. Suppose that a data block d_* needs to be inserted after the k^{th}data block d_k. First of all, the user sends d_* to CSP. Then, CSP generates auxiliary calculation information Ξ_* corresponding to it. Next, the user calculates the homomorphic tag σ_* of the data blocks according to the returned Ξ_*. Finally, the user constructs an update request Update = (Insert, $k\underline{\ }^*, \sigma_*$) and sends it to the client.

After receiving the request, CSP performs the following operations:

(1) Finds the corresponding position and inserts this data; verifies and stores the signature σ_*.

(2) Updates the MHT to get a new root hash value H'_R according to Update. Afterwards, the proof UP = $\{\text{MHT}_{\text{old}}, H(d_*), \sigma_{H_R}, H'_R\}$ is sent to the client. After receiving the proof, TPA verifies the authenticity of σ_{H_R} according to MHT_{old} and further verifies the authenticity of H'_R through $\{H(d_*), \text{Update} = (\text{Insert}, k\underline{\ }^*, \sigma_*)\}$. After verification, TPA responds with "accept" to the user. Hereafter, the user signs H'_R and sends $\sigma_{H'_R}$ to CSP. After the signature being passed, the user can delete $\{d_*, \sigma_{d_*}\}$ locally.

(2) Data Block Deletion. Suppose that the $k$thdata block d_k needs to be deleted. Firstly, the user constructs Update = (Delete, k) and sends it to CSP. After receiving the request, CSP directly finds this data block to delete, then updates MHT, and generates a new root hash value H'_R. After finishing, the update proof UP = $\{\text{MHT}_{\text{old}}, \sigma_{H_R}, H'_R\}$ will be sent to the client. After receiving it, TPA verifies the authenticity of σ_{H_R} according to MHT_{old} and further verifies the authenticity of H'_R through Update = (Delete, k). After the verification is passed, TPA responds to the user with "accept." Finally, the user signs H'_R and sends $\sigma_{H'_R}$ to CSP.

(3) Data Block Modification. Suppose that the user wants to modify the kth data block d_k to d_*. Firstly, the user sends d_* to CSP for Ξ_*. Then, the user calculates σ_* according to Ξ_*. Finally, the user constructs Update = (modify, k, $\{\underline{\ }^*, \sigma_*\}$) and sends it to CSP. After receiving the request, CSP directly replaces the original data block at the corresponding position. Then, MHT will be updated and a new root hash value H'_R will be generated. After finishing, CSP sends UP = $\{\text{MHT}_{\text{old}}, d_*, \sigma_{H_R}, H'_R\}$ to the client. After receiving it, TPA verifies σ_{H_R} according to MHT_{old} and further verifies H'_R through $\{H(d_*), \text{Update} = (\text{modify}, k, \{\underline{\ }^*, \sigma_*\})\}$. If passed, TPA responds to the user with "accept." Then, the user signs H'_R and sends $\sigma_{H'_R}$ to CSP. Finally, the user can delete $\{d_*, \sigma_{d_*}\}$ locally.

6.2.5. Batch Audit. Due to the aggregation characteristics of bilinear pairs in our scheme, TPA can audit the data of multiple users at one time, and CSP can also process multiple verification requests at the same time, which will reduce a certain cost.

7. Evaluation

In this section, we will show the correctness analysis, security analysis, performance analysis of our scheme.

7.1. Correctness Analysis. For each challenge and its corresponding proof, each entity can ensure that the data is stored correctly by verifying the equation. We will prove the correctness of these equations.

After receiving the auxiliary calculation information, the data block tag can be generated by the user based on it, which greatly simplifies the computation overhead on the user side.

The correctness of equation (1) can be proved as follows:

$$\begin{aligned}\sigma_i &= H(d_i)^{\alpha_u} \cdot v^{d_i} \\ &= H(d_i)^{\alpha_u} \cdot \left(g^{\alpha_u}\right)^{d_i} \\ &= H(d_i)^{\alpha_u} \cdot \left(g^{d_i}\right)^{\alpha_u} \\ &= \left(H(d_i) \cdot g^{d_i}\right)^{\alpha_u}.\end{aligned} \tag{7}$$

The correctness of equation (2) can be proved as follows:

$$\begin{aligned} e\left(\sigma_{H_R}, g\right) &= e\left(H_R^{\alpha}, g\right) \\ &= e\left(H_R, g^{\alpha}\right) \\ &= e\left(H_R, v\right). \end{aligned} \tag{8}$$

In the data outsourcing stage, the correctness of equation (3) can be proved as follows:

$$\begin{aligned} e\left(\sigma_i, g\right) &= e\left(\left(H\left(d_i\right) \cdot g^{d_i}\right)^{\alpha}, g\right) \\ &= e\left(H\left(d_i\right) \cdot g^{d_i}, g^{\alpha}\right) \\ &= e\left(H\left(d_i\right) \cdot g^{d_i}, v\right). \end{aligned} \tag{9}$$

Then, the proof (6) can be proved as follows:

$$\begin{aligned} e\left(\sigma, g\right) &= e\left(\prod_{\left(I_i, r_i\right) \in Cl} \sigma_i^{r_i}, g\right) \\ &= e\left(\prod_{\left(I_i, r_i\right) \in Cl} \left(H\left(d_i\right) \cdot g^{d_i}\right)^{\alpha_u r_i}, g\right) \\ &= e\left(\prod_{\left(I_i, r_i\right) \in Cl} \left(H\left(d_i\right)^{r_i} \cdot g^{r_i d_i}\right)^{\alpha_u}, g\right) \\ &= e\left(\prod_{\left(I_i, r_i\right) \in Cl} \left(H\left(d_i\right)^{r_i} \cdot g^{r_i d_i}\right), g^{\alpha_u}\right) \\ &= e\left(\prod_{\left(I_i, r_i\right) \in Cl} \left(H\left(d_i\right)^{r_i} \cdot g^{\Re}\right), v\right). \end{aligned} \tag{10}$$

Due to the aggregation feature of bilinear pairs, all the proof can be verified at once, which greatly reduces the burden of the audit.

7.2. Security Analysis

Lemma 1. *CSP cheats during the auxiliary information generation; it will be checked out with a nonnegligible probability.*

Proof. CSP cheats during the process of generating auxiliary information. At this time, there are two situations: (1) after data uploading, in order to save storage space, CSP deliberately discards part of data and selects saved data blocks of other users to perform auxiliary calculation information Ξ so as to trick the user into real tag generation of the fake data block to pass the follow-up audit. Let a be the ratio of error hashes, b the probability that the user detects an error, and x the number of hashes that the user needs to detect, then$b = 1 - (1 - a)^x$. So, the user only needs to calculate 460 hashes to detect them with a 99% probability. (2) CSP performs an honest calculation of the user's MHT but forges the calculation of v^{d_i}; then in the follow-up audit, equation (3) cannot be passed because of

$$\begin{aligned} e\left(\sigma_i, g\right) &= e\left(H\left(d_i\right)^{\alpha_u} \cdot v^{d_i}, g\right) \\ &\neq e\left(H\left(d_i\right)^{\alpha_u} \cdot g^{\alpha_u d_i}, g\right) \\ &= e\left(H\left(d_i\right) \cdot g^{d_i}, v\right). \end{aligned} \tag{11}$$

In both situations, it is impossible for CSP to pass the verification. □

Lemma 2. *CSP loses part of the data; it cannot respond to the audit challenge successfully.*

Proof. Without loss of generality, suppose that d_{11} in the challenged data block has been lost; there are three situations at this time: (1) CSP hopes to pass the audit by forging this data, that is, forging d_{11} as d'_{11}. Then, due to the change of node hash, the root hash changes accordingly, equation (2) for root node signature verification is invalid, finally, the challenge response fails, and the CSP cannot pass the audit. (2) CSP hopes to replace d_{11} with other data blocks d_{21} of the user to participate in the challenge response; then the auxiliary authentication information changes, the root hash changes accordingly, equation (2) does not hold, and the challenge response fails. (3) CSP retains the hash value of it, equation (2) can be passed at this time, but the hash is irreversible, and CSP cannot reversely obtain d_{11}, so it cannot pass the verification of (6). Therefore, if the CSP loses part of the data, it will not be able to successfully cope with the challenge. □

Lemma 3. *CSP and TPA cannot conduct collusion attacks.*

Proof. CSP and TPA conduct a collusion attack; there are two situations at this time. (1) TPA deliberately selects data blocks that are intact in CSP to challenge. (2) TPA forged the correctness of audit results. Aiming at 1, our scheme uses the Ethereum blockchain to generate challenge request automatically, and the selected challenge blocks with their corresponding coefficients are randomly generated. Therefore, it is impossible to manually select the challenge number and its corresponding coefficient. So, TPA cannot help CSP to falsify audit results by deliberately selecting complete blocks. For 2, due to the existence of user sampling audit, if TPA falsifies the audit results, it cannot pass it. Therefore, our scheme can avoid collusion attacks. □

Lemma 4. *TPA does not respond to the challenge in time; it will be detected by the user.*

Proof. TPA does not perform the audit within the prescribed time limit but audits all previous data at one time before the user audit, which will make the audit invalid, because the missing data blocks cannot be detected in time. Due to the introduction of the blockchain, TPA needs to send 0eth to the user's contract account after each audit and mark the challenge-response data block, so that each audit process is public and verifiable. And because of the existence of user audit, delayed audits will be detected in time.

Therefore, it can be detected whether the audit was performed by the TPA in time. □

7.3. Performance Analysis. In this section, we analyse the computation and memory overhead, and we make a brief comparison with Rao's [29].

This scheme mainly involves four basic cryptographic operations: hash operation, modular exponentiation operation, modular multiplication operation, and bilinear pair operation, which are represented by symbols T_H, T_{exp}, T_{mul}, and T_{bp}, respectively. We concluded these basic operations at each stage according to the scheme in Table 2.

We compared the computation overhead of user, CSP, and TPA at various stages in Table 2. It can be seen intuitively that our computation overhead on the user side has a great advantage. Because we outsource part of the calculation to CSP to reduce the user's computation overhead, this is especially friendly to lightweight IoT devices.

Besides, we comprehensively evaluate our performance through experiments. The experimental environment settings are as follows: the laptop hardware configuration information is 3.30 GHz Core i5-4590 CPU, 4 GB DDR3-1600 RAM. Each algorithm uses the JAVA programming language and uses the currently popular JAVA cryptography library JPBC 2.0 which integrates a large number of commonly used cryptography algorithms. At the same time, we choose the D-type curve to construct the group G, and the group element size is set to 21 bytes, the large prime q is 1024 bits, and the hash operation is SHA-256. The outsourcing data set is set to 1 GB, and the data block size is set to 1 kB to 256 kB, respectively.

Firstly, we measured the computation overhead of the user side during the data outsourcing stage. Figure 1 shows the computation overhead of the user side changes with the block size in the data outsourcing stage. The abscissa represents the data block size, and the ordinate represents the user-side computation time. We can intuitively see that, as the data block size increases, that is, the number of blocks decreases, the computation overhead of both schemes decreases. The computation time difference reaches the maximum when the data block size is the smallest and reaches the minimum when the data block size is the largest. This is because the more the number of blocks, the more tags that need to be calculated, the greater the depth of the constructed MHT, and the greater the overhead. That is to say, in the case of more data blocks, the computational overhead of our solution has a greater advantage. This advantage comes from the fact that we transfer most of the calculations to CSP with higher computational performance. Therefore, when data outsourcing, our scheme is more friendly to lightweight devices with weak storage and computing capabilities.

Then, we measured the computation overhead of the user side during the dynamic update stage. Figure 2 shows the computation overhead of the user side changes with the block size in the dynamic update stage. The abscissa represents the data block size, and the ordinate represents the user-side computation time. We can see intuitively that, at this time, the computation overhead of our scheme is a small constant value and does not change with the change of the block size. The computation overhead of Rao's [29] decreases as the block size increases. The main reason is that in the dynamic update stage, data block insertion, deletion, or modification operations will cause changes in the structure of the Merkle hash tree. At this time, a large number of node information in the hash tree will change accordingly, which will cause a lot of recalculation in the whole process. Therefore, as the number of blocks increases, the number of recalculations increases. Our scheme fully demonstrates the advantages of computation overhead, and users only need very little verification calculation.

Finally, we measured the communication overhead of the user side during the data outsourcing stage. We can intuitively see from Figure 3 that, as the size of the data block increases, that is, the number of data blocks decreases, the communication overhead of both schemes gradually decreases and tends to 1 GB. At this stage, as the data is finally uploaded to the CSP, the communication overhead of the 1 GB data is indispensable. In addition, it also involves the upload of some auxiliary information, such as data tags and Merkle hash tree. Our scheme reduces the process of uploading part of the auxiliary information to the TPA, thereby reducing part of the communication overhead, so our scheme has a slight advantage at this stage. We believe that authorized third party should retain very little user data information, and at the same time, any authorized entity can perform audits on behalf of users when they only know the user's public key information, realizing a true public audit.

In summary, due to the outsourcing of part of the calculation, our scheme significantly reduces the computation overhead on the user side. Since the data is ultimately stored in the CSP, selecting CSP instead of TPA for some calculations will reduce overhead to a certain extent.

Table 2: Comparison of computation overhead for different entities at different stages.

Stage	entity	Data outsourcing	TPA audit	Insert	Delete	Modify
User side	Our	$460T_H + (n+2)T_{\text{exp}} + nT_{\text{mul}}$	0	$1T_H + 2T_{\text{exp}} + 1T_{\text{mul}}$	$1T_{\text{exp}}$	$1T_H + 2T_{\text{exp}} + 1T_{\text{mul}}$
	Rao's	$(2n-1)T_H + (3n+1)T_{\text{exp}} + nT_{\text{mul}}$	0	$(\log_2 n + 2n + 3)T_H + 4T_{\text{exp}} + 1T_{\text{mul}} + 1T_{\text{bp}}$	$(\log_2 n + 2n)T_H + 1T_{\text{exp}} + 1T_{\text{bp}}$	$(\log_2 n + 2n + 1)T_H + 4T_{\text{exp}} + 1T_{\text{mul}} + 1T_{\text{bp}}$
CSP side	Our	$(2n-1)T_H + 2nT_{\text{exp}} + nT_{\text{mul}} + (n+1)T_{bp}$	$sT_{\text{exp}} + 2sT_{\text{mul}}$	$(\log_2 n + 3)T_H + 1T_{\text{exp}} + 2T_{\text{bp}}$	$(\log_2 n)T_H + 1T_{\text{bp}}$	$(\log_2 n + 1)T_H + 1T_{\text{exp}} + 2T_{\text{bp}}$
	Rao's	$(2n-1)T_H + (n+1)T_{\text{bp}}$	$sT_{\text{exp}} + 2sT_{\text{mul}}$	$(\log_2 n + 3)T_H + 2T_{\text{bp}}$	$(\log_2 n)T_H + 2T_{\text{bp}}$	$(\log_2 n + 1)T_H + 2T_{\text{bp}}$
TPA side	Our	0	$460T_H + (s+1)T_{\text{exp}} + (s+1)T_{\text{mul}} + 2T_{\text{bp}}$	$(\log_2 n + 463)T_H + 1T_{\text{bp}}$	$(\log_2 n + 460)T_H + 1T_{\text{bp}}$	$(\log_2 n + 461)T_H + 1T_{\text{bp}}$
	Rao's	$(2n-1)T_H$	$(s+2)T_{\text{exp}} + (s+2)T_{\text{mul}} + 1T_{\text{bp}}$	$(\log_2 n + 3)T_H$	$(\log_2 n)T_H$	$(\log_2 n + 1)T_H$

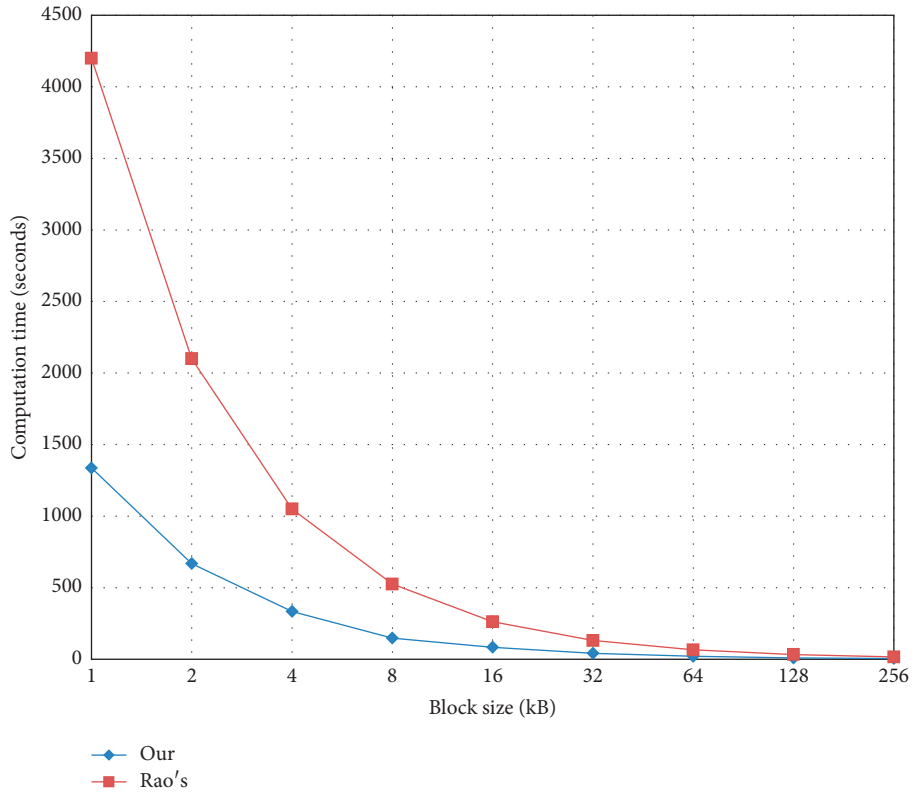

Figure 1: User-side computation overhead varies with the data block size in the data outsourcing stage.

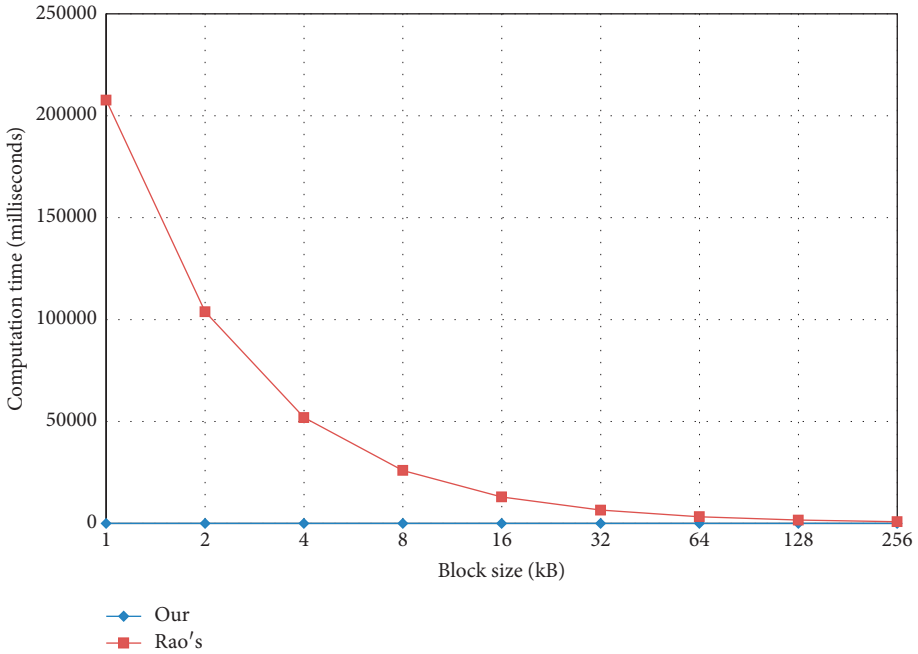

Figure 2: User-side computation overhead varies with the data block size in the dynamic update stage.

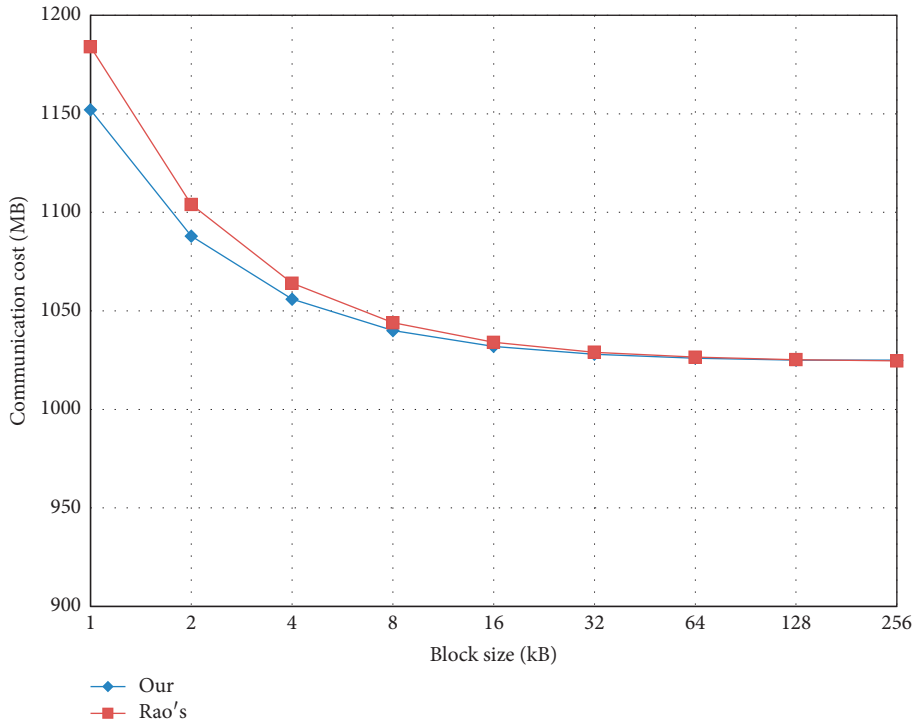

Figure 3: The communication overhead varies with the block size in the data outsourcing stage.

8. Conclusions

We have conducted research on outsourcing audit under the IoT environment and designed an audit protocol suitable for lightweight devices, reducing the computing burden of the device by transferring part of the computing overhead to CSP. The challenge request is generated through the Ethereum blockchain to avoid collusion attacks. Therefore, our scheme has higher security. The experimental results show that our scheme has higher efficiency.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key Program of NSFC (no. U1405255), the Shaanxi Science and Technology Coordination and Innovation Project (no. 2016TZC-G-6-3), the National Natural Science Foundation of China (nos. 61702404 and 61702105), the China Postdoctoral Science Foundation Funded Project (no. 2017M613080), the Fundamental Research Funds for the Central Universities (nos. JB171504 and JB191506), the National Natural Science Foundation of Shaanxi Province (no. 2019JQ-005), and the Graduate Innovation Foundation of Xidian University (no. 20109194858).

References

[1] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Information Sciences*, vol. 494, pp. 193–207, 2019.

[2] Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang, "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system," *Information Sciences*, vol. 479, pp. 567–592, 2019.

[3] X. Liu, R. H. Deng, Y. Yang, H. N. Tran, and S. Zhong, "Hybrid privacy-preserving clinical decision support system in fog-cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 825–837, 2018.

[4] Y. Miao, X. Liu, K. K. R. Choo et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[5] Y. Miao, R. Deng, X. Liu et al., "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[6] Y. Miao, R. Deng, K. K. R. Choo et al., "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[7] Gartner.com, Gartner Says Worldwide IaaS Public Cloud Services Market Grew 37.3% in 2019, 2020, https://www.gartner.com/en/newsroom/press-releases/2020-08-10-gartner-says-worldwide-iaas-public-cloud-services-market-grew-37-point-3-percent-in-2019.

[8] Amazon.com, Amazon S3 Availability Event: July 20, 2008, http://status.aws.amazon.com/s3-20080720.html.

[9] G. Hou, J. Ma, C. Liang et al., "Efficient audit protocol supporting virtual nodes in cloud storage," *Transactions on Emerging Telecommunications Technologies*, Article ID e3911, 2020.

[10] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor, "Checking the correctness of memories," *Algorithmica*, vol. 12, no. 2-3, pp. 225–244, 1994.

[11] A. Juels and B. S. Kaliski, "PORs: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 584–597, Alexandria, VA, USA, October 2007.

[12] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90–107, Melbourne, Australia, December 2008.

[13] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[14] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, Alexandria, VA, USA, November 2007.

[15] G. Ateniese, R. D. Pietro, L. V. Mancini et al., "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, pp. 1–10, Istanbul, Turkey, September 2008.

[16] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security*, vol. 17, no. 4, pp. 1–29, 2015.

[17] C. Wang, Q. Wang, K. Ren et al., "Ensuring data storage security in Cloud Computing," in *Proceedings of the 2009 17th International Workshop on Quality of Service*, pp. 1–9, Charleston, SC, USA, July 2009.

[18] C. Wang, Q. Wang, K. Ren et al., "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, San Diego, CA, USA, March 2010.

[19] Q. Wang, C. Wang, K. Ren et al., "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2010.

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

[21] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717–1726, 2012.

[22] F. Armknecht, J. M. Bohli, G. O. Karame et al., "Outsourced proofs of retrievability," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 831–843, Scottsdale, AZ, USA, November 2014.

[23] C. Liu, R. Ranjan, C. Yang et al., "MuR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2014.

[24] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in *Proceeding of the 20th European Symposium on Research in Computer Security*, pp. 203–223, Vienna, Austria, September 2015.

[25] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, 2016.

[26] M. Sookhak, F. R. Yu, and A. Y. Zomaya, "Auditing big data storage in cloud computing using divide and conquer tables," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 999–1012, 2017.

[27] G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das, and J. J. P. C. Rodrigues, "SecSVA: secure storage, verification, and auditing of big data in the cloud environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78–85, 2018.

[28] W. Shen, J. Qin, J. Yu et al., "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2018.

[29] L. Rao, H. Zhang, and T. Tu, "Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 451–463, 2017.

[30] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proceeding of the 21st Annual International Cryptology Conference*, pp. 213–229, Santa Barbara, CA, USA, August 2001.

[31] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[32] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," in *Proceedings of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, pp. 224–314, Paris, France, April 1984.

[33] R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the IEEE Symposium on Security and Privacy*, p. 122, Oakland, CA, USA, April 1980.

[34] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system, Manubot," Technical Report, 2019.

[35] N. Singh and M. Vardhan, "Distributed ledger technology based property transaction system with support for IoT devices," *International Journal of Cloud Applications and Computing*, vol. 9, no. 2, pp. 60–78, 2019.