mathematics

Article

A Blockchain-Empowered Arbitrable Multimedia Data Auditing Scheme in IoT Cloud Computing

Shenling Wang, Yifang Zhang and Yu Guo *

School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China; slwang@bnu.edu.cn (S.W.); zyfyydgq@mail.bnu.edu.cn (Y.Z.)
* Correspondence: yuguo@bnu.edu.cn

Abstract: As increasing clients tend to outsource massive multimedia data generated by Internet of Things (IoT) devices to the cloud, data auditing is becoming crucial, as it enables clients to verify the integrity of their outsourcing data. However, most existing data auditing schemes cannot guarantee 100% data integrity and cannot meet the security requirement of practical multimedia services. Moreover, the lack of fair arbitration leads to clients not receiving compensation in a timely manner when the outsourced data is corrupted by the cloud service provider (CSP). In this work, we propose an arbitrable data auditing scheme based on the blockchain. In our scheme, clients usually only need to conduct private audits, and public auditing by a smart contract is triggered only when verification fails in private auditing. This hybrid auditing design enables clients to save audit fees and receive compensation automatically and in a timely manner when the outsourced data are corrupted by the CSP. In addition, by applying the deterministic checking technique based on a bilinear map accumulator, our scheme can guarantee 100% data integrity. Furthermore, our scheme can prevent fraudulent claims when clients apply for compensation from the CSP. We analyze the security strengths and complete the prototype's implementation. The experimental results show that our blockchain-based data auditing scheme is secure, efficient, and practical.

Keywords: data auditing; data integrity; bilinear map accumulator; blockchain; smart contract

MSC: 68P20

check for updates

Citation: Wang, S.; Zhang, Y.; Guo, Y. A Blockchain-Empowered Arbitrable Multimedia Data Auditing Scheme in IoT Cloud Computing. *Mathematics* **2022**, *10*, 1005. https://doi.org/10.3390/math10061005

Academic Editors: Ximeng Liu, Yinbin Miao and Zuobin Ying

Received: 8 February 2022
Accepted: 17 March 2022
Published: 21 March 2022

1. Introduction

The rapid development of the Internet of Things (IoT) and intelligent multimedia has led to the explosive growth of massive amounts of data, which has put tremendous pressure on the entire Internet. To cope with this challenge, storing IoT and intelligent multimedia data with a cloud service provider (CSP) is a common solution [1–4]. Many schemes have been implemented and proved to ensure the security of outsourced data during transmission [5]. However, such a wide attack surface and many recent data breaches have raised concerns about data integrity and availability [6–10]. When sensitive IoT and intelligent multimedia data are outsourced to a CSP, the clients lose control of the data, and the data may be changed or deleted without their permission. To solve this problem, clients need to regularly check the integrity of the outsourced data, and remote data integrity checking is becoming an important issue in cloud computing.

Most existing data integrity checking techniques are probabilistic [11–27]. In this approach, the verifier randomly selects partial data blocks and then performs integrity verification on those chosen data blocks instead of checking the whole dataset, and hence a 100% guarantee for the integrity of the data cannot be provided. However, for massive IoT and intelligent multimedia data, especially sensitive data related to finance, energy, transportation, etc., a probabilistic approach is not enough, since they have strict requirements for data integrity and correctness. Another type of data integrity checking technique is deterministic. In this approach, the verifier examines all data blocks instead of only checking chosen partial data blocks, thus providing 100% assurance of data integrity [28]. However, the deterministic method means higher verification and computational overhead, and hence efficiency is a challenge and must be considered in this approach.

In order to check remote outsourced data integrity, numerous data auditing schemes have been proposed. According to the different roles of verifiers, the existing auditing schemes can be divided into private auditing and public auditing [24]. In private auditing schemes, the role of the verifier is assumed by the client himself, and some important key information used in verification is usually stored by the client instead of the CSP. Therefore, there will be disputes when the verification fails, because if the key information used in the verification is broken by a malicious client, the response of the CSP cannot pass verification. In other words, we cannot determine whether the CSP has damaged the data when verification fails. At this point, fair arbitration is required because the CSP needs to compensate the client for data corruption if verification is not passed. In public auditing schemes, the client usually resorts to a third-party auditor (TPA) to check the integrity of outsourced data. Thus, the audit results completely depend on the TPA. However, this is unrealistic since a fully credible TPA may not always exist. In addition, it should be noted that in these two existing types of auditing schemes, if the auditing results show that the outsourced data are corrupted by the CSP, it is usually difficult for clients to obtain the compensation from the CSP in a timely manner.

In this work, we propose an efficient blockchain-based hybrid auditing scheme with fair arbitration. In our scheme, we use bilinear map accumulators to realize deterministic checking, in which the verifier can check all data blocks, and at the same time, the computational overhead is acceptable. Specifically, for outsourced data $\mathrm{B} = \{b_1, \cdots, b_n\}$, the basic idea of an audit is that the verifier uses the random index j to challenge the CSP. Upon receiving the challenge, the CSP needs to compute the corresponding witness wit_{b_j} for the target data block b_j, and all data blocks except b_j are used in the calculation of the witness. The CSP returns both the target data block b_j and the calculated witness as a response, namely (wit_{b_j}, b_j). Thus, even a small change in the outsourced data can cause the generated response to change, and so the generated response cannot pass verification. In other words, a valid response cannot be generated by the CSP if the data are not actually saved or the data are corrupted with the CSP. Therefore, the verifier examines all data blocks instead of only checking partial data blocks, thus providing 100% assurance of data integrity.

Moreover, the client not only holds the data file digest acc_B, which is the key information used in verification by him or herself, but also saves a copy of the digest acc_B to the blockchain simultaneously with the data uploading phase. During the audit, the role of the verifier can be assumed by a client or blockchain smart contract, which means that clients usually only need to conduct private audits, and public auditing by a smart contract is triggered only when verification fails in private auditing because if the data file digest is broken by the client, the CSP's response cannot pass verification even if the data are not corrupted by the CSP, and disputes may arise in this point. In this case, it will trigger the blockchain smart contract to conduct the public auditing for fair judgement, since the data file digest acc_B saved in the blockchain will not be broken by anyone due to the non-tamperability property of the blockchain. Therefore, this blockchain-based hybrid auditing scheme with fair arbitration can solve the problem of distrust between the CSP and the client.

It is worth noting that we replace the TPA with a blockchain smart contract in our public auditing phase, and by using this technology, designing a smart contract with fair arbitration can ensure that the client will be compensated automatically and in a timely manner when the outsourced data are corrupted by the CSP due to the smart contract, which is an automatically executed code running on the blockchain. Furthermore, a dishonest client may falsely claim for compensation from the CSP, and we resort to digital signature technology to prevent dishonest behavior by the client. The contributions of our work are summarized as follows:

- We present an efficient hybrid data auditing scheme for the IoT and intelligent multimedia by using the blockchain. By applying deterministic cryptographic techniques and the blockchain, our proposed design can fairly solve the problem of distrust between the CSP and the client. It also makes the auditing scheme more reliable, because the deterministic methods provide 100% data possession and integrity guarantees.
- We enforce a healthy ecosystem to punish dishonest CSPs automatically and provide timely compensation to the client for data corruption by the CSP in the proposed scheme.
- Our scheme can protect honest CSPs and prevent fraudulent claims by dishonest clients at the same time.
- We use the hybrid auditing design in the proposed scheme. It can also save audit fees and communication costs for the client, because the public auditing phase is triggered only when verification fails in the private auditing phase.
- We not only theoretically prove the correctness and soundness of our scheme but also experimentally verify the feasibility and efficiency of the scheme by the prototype's implementation.

The rest of the paper is structured as follows. Section 2 overviews some related works, and Section 3 introduces the preliminaries used in our scheme. After that, we give the system model of our proposed design, including the architecture overview, threat model, and security goals, in Section 4. Section 5 presents a detailed description of our proposed scheme. In Section 6, we present security analysis and some characteristics of our proposed scheme. We show the performance evaluation in Section 7 and conclude the paper in Section 8.

2. Related Works

2.1. Data Auditing Schemes

With the increase in demand for outsourced data integrity checking, many data auditing schemes have been proposed [29,30]. Ateniese et al. [11] introduced the notion of provable data possession (PDP), which was the first public audit scheme to verify the authenticity of data, in 2007. However, data privacy protection and the full data dynamic operation cannot be supported in this scheme [12]. Then, Erway et al. [13] proposed a PDP scheme supporting full dynamic data updating. Since then, to achieve more functions and improve the efficiency of data auditing for remote data, a lot of research has been conducted in this area. Wang et al. [14] proposed a public data auditing scheme supporting data dynamic operations. In [15], the follow-up work supports privacy-preserving multiple-task auditing. Yuan et al. [16] proposed a public audit scheme for dynamic data sharing with the help of doubly linked information tables. In [17], the authors used the data structure of a Merkle hash tree to devise a public auditing scheme in which the communication overhead and verification efficiency are greatly taken into account. In addition to public auditing, private auditing is necessary in some cases [21–24]. Furthermore, various PDP models have been proposed [25–27]. The PDP method above allows a verifier to verify the remote data integrity without retrieving or downloading all of the data, only randomly selecting a few data blocks and then performing integrity verification on those chosen data blocks instead of checking the whole dataset. Thus, this is a probabilistic method and cannot provide a 100% guarantee for the data's integrity.

Due to the limitations of hardware, few deterministic auditing schemes have been proposed. In [31], the authors proposed the first deterministic public auditing mechanism, but it did not support dynamic data operation. In [32], Deswarte et al. proposed an auditing scheme based on the Diffie–Hellman cryptographic protocol. However, it incurred a high computational overhead, because the CSP must compute the power of the entire file for each auditing verification. Filho et al. [33] proposed a simple deterministic data integrity checking protocol based on a homomorphic RSA-based hash function, but the computation cost remained high and without data dynamic support. In [34], Sebé et al. devised a data possession checking protocol based on the Diffie–Hellman key exchange, which can reduce the computational overhead but without public auditing or data dynamic support. Barsoum [35] proposed a multi-copy provable data possession scheme supporting the public verifiability of multiple replicas of the data. In [36], Hao et al. proposed a privacy-preserving data integrity auditing scheme that supports public auditing and data dynamics.

With the development of the IoT and intelligent multimedia, several data auditing schemes for the IoT and intelligent multimedia services have been proposed. In [28], the authors devised a data audit mechanism by using a bilinear mapping accumulator for sensor data. The proposed design can check all data blocks, thereby eliminating the possibility of any server-side operation. However, existing data auditing solutions cannot solve the trust issue between the data owner and the CSP (or TPA). For private auditing schemes, since the key information used in verification is stored only on the client's side locally, this cannot solve the problem of client fraud. For public auditing schemes, the data owner usually resorts to a TPA to check the outsourced data integrity, but this is unrealistic as a fully trusted TPA may not always exist. As mentioned in [18], the involvement of a TPA may lead to data loss or abuse of authority.

2.2. Blockchain and Smart Contracts

As the core technology of the emerging cryptocurrencies, the blockchain is essentially a distributed database where the transactions are batched into an ordered growing list of blocks which are linked using cryptography. As is well known, the blockchain has the characteristics of decentralization, immutability, and distributed storage. Smart contracts [37] are executable, pre-agreed programs running automatically on the blockchain. Based on the properties and functions of the blockchain, many typical applications such as decentralized storage [38–41], crowdsourcing systems [42–45], medical data management [46,47], and distributed ledger technologies [48,49] have been built.

Recently, Wang et al. [19] leveraged smart contracts to design a blockchain-based fair payment scheme to replace TPAs for public cloud auditing. Yuan et al. [20] proposed a blockchain-based public auditing and secure deduplication scheme which supports automatic compensation of users for data corruption and automatic punishment of malicious CSPs by using a smart contract, but users need to pay an audit fee to the miners of the blockchain in each verification. Wang et al. [24] proposed a blockchain-based private provable data possession scheme which not only saves storage space but also greatly improves efficiency. However, it has no mechanism for automatic punishment and compensation when the outsourced data are corrupted by a CSP. Moreover, theses blockchain-based auditing schemes are all for the probabilistic approach. In light of the previous work, our work used a different auditing design which combines private auditing and public auditing for a deterministic approach. We aim to ensure reliability for data integrity verification and financial fairness in the data auditing scheme so that both the clients and CSPs are incentivized to conduct trustworthy behavior while saving on auditing fees for the clients. Table 1 shows the comparison between our proposed design and some related existing auditing schemes.

Table 1. Comparison with some related existing data integrity auditing schemes.

Scheme	**Wang et al. [19]**	**Yuan et al. [20]**	**Ren et al. [28]**	**Wang et al. [24]**	**Our Scheme**
Deterministic audit	×	×	√	×	√
Fair arbitration	√	√	×	√	√
Without third-party auditor	√	√	×	√	√
Data dynamic supporting	×	×	√	×	√
Privacy preserving	√	√	√	√	√
Automatic compensation and punishment mechanism	√	√	×	×	√

3. Preliminaries

3.1. Bilinear Mapping

Let G_1, G_2, and G_T be three cyclic groups of the prime order p. We use g_1 and g_2 to denote the generator of G_1 and G_2, respectively. Bilinear mapping (pairing) is a mapping $e: G_1 \times G_2 \rightarrow G_T$ with the following properties:

- Bilinearity: $\forall$ x, y $\in Z_p$, $e({g_1}^x, {g_2}^y) = e(g_1, g_2)^{xy} = e({g_1}^y, {g_2}^x)$;
- Non-degeneracy: $e(g_1, g_2) \neq 1_{G_T}$; that is, $e(g_1, g_2)$ generates G_T;
- Computability: For all x, y $\in Z_p$, there exists an effectively computable algorithm to compute $e({g_1}^x, {g_2}^y)$.

3.2. q-SDH Assumption

Here we assume that $G_1 = G_2 = G$. Therefore, let G be a finite cyclic group of the order p, where p is a prime number whose length is κ bits. Thus, for a randomly chosen element $\alpha \in Z_p^*$, a random generator g of G, and PPT algorithm A, the following holds:

$$\Pr\left[(c,\ g^{\frac{1}{\alpha+c}}) \leftarrow A(g,\ g^{\alpha},\ \cdots\cdots,\ g^{\alpha^q})\right] \leq \epsilon(\kappa) \text{ for some } c \in Z_p \backslash \{-\alpha\}$$

where ϵ (κ) denotes a negligible function.

The bilinear map accumulator to be used in our data integrity auditing scheme is based on the properties of bilinear mapping and the q-SDH assumption described above. See [28] for details.

3.3. Smart Contract

The concept of a smart contract was first proposed by Nick [37] in 1995. It is an executable pre-agreed program running automatically on the blockchain according to its content. Developers can build distributed applications such as voting, financial transactions, and signing agreements based on smart contracts for Ethereum. When deploying the smart contract, it is necessary to preset the trigger condition and the corresponding response rule. After the smart contract is deployed, once an event triggers the terms of the contract, the code will be executed automatically without central authorization. The relevant details can be found in [37].

4. System Model

We present the system model, threat model, and security goals of our proposed hybrid auditing scheme with fair arbitration for data integrity verification in this section.

4.1. Architecture Overview

The traditional cloud data integrity auditing scheme consists of three roles—the client, verifier, and CSP—as shown in Figure 1. The client is the data owner who wants to outsource their personal data to a CSP, the CSP provides outsourced data storage and management services for the client, and the verifier is in charge of auditing the outsourced data's integrity. The role of the verifier can be performed by a client or a third-party auditor (TPA), which correspond to private auditing and public auditing, respectively. Our proposed blockchain-based hybrid auditing scheme extends the work performed in [20,28]. The integrity verification scheme in [28] meets both private and public auditing, but both of these types of audits have some defects: (1) In private auditing, when verification fails, we cannot determine whether the CSP has damaged the data because the file digest acc_B used in verification is stored on the client's side locally. If the file digest is broken by the client, the CSP's response cannot pass verification. At this point, fair arbitration is required, because the CSP needs to compensate the client for data corruption in this case. (2) In public auditing, the client resorts to a TPA to audit the data's integrity. However, it is unrealistic that the correctness of the auditing results depends entirely on the TPA.

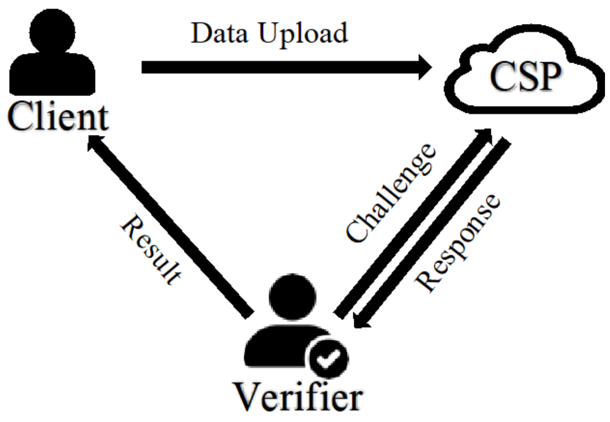

Figure 1. Overview of data integrity auditing.

Based on the defects in above two types of auditing in [28], we propose an arbitrable hybrid auditing scheme based on the blockchain. In this scheme, the client first conducts a private audit, and in case verification fails, it will trigger the blockchain to perform a public audit for fair judgement. It is worth noting that the TPA is replaced by the blockchain in public audits in our scheme, and we designed a smart contract with fair arbitration for the client. Using this smart contract, the client can be compensated automatically when data are broken by the CSP. Our scheme includes three different roles: the client, CSP, and blockchain. As shown in Figure 2, the interactions among them are described as follows:

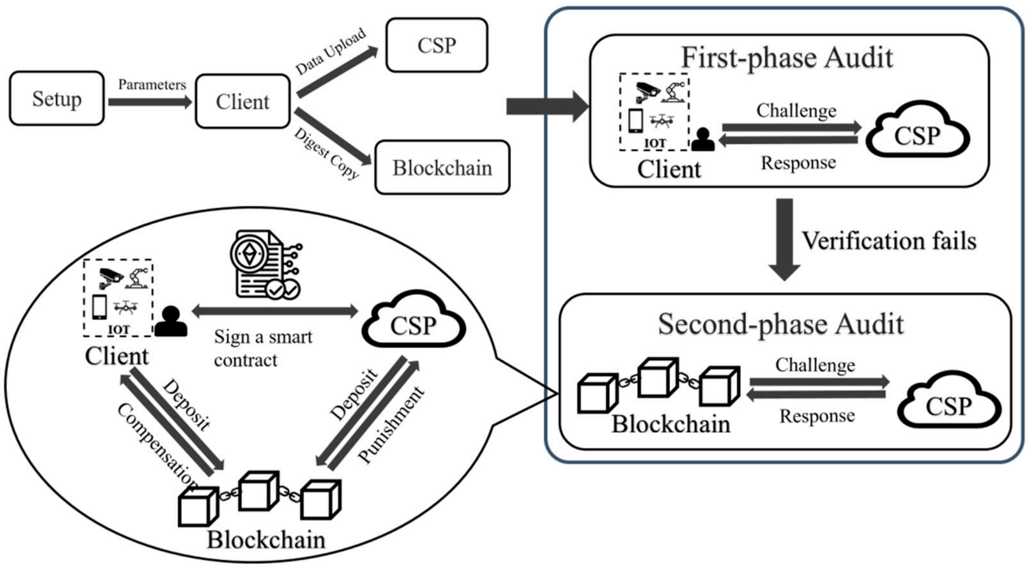

Figure 2. Architecture of our two-phase data auditing scheme.

1. The client has a large amount of data and needs to outsource this data to the CSP for maintenance and computation but does not save the copy locally. It also stores a file digest copy on the blockchain. Then, the client challenges the CSP and verifies the response coming from the CSP. If the response from the CSP passes verification, the outsourced data are considered complete; otherwise, they are considered incomplete. This case will trigger the blockchain to perform public auditing for fair judgement.
2. The CSP has huge storage space and computation resources to provide outsourced data storage and management services for the client. Upon receiving the client's (or blockchain's) challenge, the CSP sends the generated response to the client (or blockchain).
3. The blockchain stores copies of the file digests for the client. When the blockchain is triggered to perform public auditing for fair judgement, the blockchain will challenge

the CSP and then verify the response coming from the CSP. If the response can pass verification, the remote data are complete; otherwise, they are determined to be incomplete, since the data file digest acc_B saved in the blockchain will not be broken by anyone due to the non-tamperability property of the blockchain. Therefore, the failure of verification must be due to damage to the outsourced data by the CSP, and the client obtains compensation from the CSP automatically through a smart contract.

4.2. Threat Model and Design Goals

Both the CSP and the clients can be dishonest in our scheme. It is assumed that the CSP has no motivation to disclose managed data to others and also has no motivation to drop the managed data. However, the data stored on the CSP may be damaged due to software, hardware bugs, or hacker attacks. The CSP may conceal data corruption to avoid compensation. For the clients, they may modify the file digests used in private auditing, which leads to a failed verification result for obtaining compensation from the CSP. Moreover, one may pass him or herself off as a real client (i.e., real data owner) to obtain compensation from the CSP.

In this scheme, we will achieve the following security goals:

- Correctness: If the outsourced data have not been broken by the CSP, and if the client and CSP execute the proposed scheme honestly, then the response from the CSP can pass verification;
- Soundness: Only when the data are complete can they pass verification;
- Privacy preserving: The entire auditing process will not disclose any data privacy in-formation;
- Dynamic operations support: This supports that the client can insert, delete, and update the data outsourced to the CSP, and after dynamic operations, the auditing scheme remains applicable;
- Timely compensation: The client can obtain compensation from the CSP in a timely manner when the outsourced data are damaged by the CSP.

In order to better understand our audit scheme's construction, the major notations and their meanings in this paper are listed in Table 2.

Table 2. Notations.

Notation	Meaning
e	A bilinear pairing
G, G_1, G_2, G_T	Cyclic groups with order p
g	Generator of group G
F	The original data file to be divided into n segments $f_1, \cdots, f_n$
c_i	The ciphertext of segment f_i
τ_i	The tag of segment c_i, i.e., $\tau_i = H(c_i)$
H	A cryptographic hash function
B	The processed data file B $= (b_1, \cdots, b_n)$ with $b_i = c_i \parallel \tau_i$
acc_B	The accumulated value of data file B by the bilinear pair accumulator
wit_{b_j}	The witness calculated by the CSP
σ	The signature of file B
ssk, spk	The private key ssk and public key spk for a digital signature algorithm
sk_{acc}, pk_{acc}	The sec ret key sk_{acc} and public key pk_{acc} for a bilinear pair accumulator

5. Scheme Construction

We first give the main idea of our design and then show the auditing scheme in detail.

5.1. Main Idea

Our proposed blockchain-based hybrid auditing scheme extends the work in [20] and [28], but the difference is that the role of the verifier can be the client or blockchain smart contract, which enables our scheme to audit with fair arbitration and timely compensation, saving audit fees and communication costs for the client.

In our proposed blockchain-based hybrid auditing scheme, to save audit fees and communication costs for the client, the audit phase is executed by the client first (i.e., the client assumes the role of the verifier for the audit). If verification is passed, then the outsourced data are complete; otherwise, the outsourced data are possibly incomplete. At this point, the blockchain smart contract is triggered for fair arbitration; that is, the blockchain smart contract assumes the role of the verifier for auditing again. Meanwhile, both the client and CSP send deposits to the smart contract, and then client signs a smart contract with the CSP. First, the smart contract submits the deposit of the client to the miners as an audit fee. Secondly, if verification is passed, the CSP's deposit is returned; otherwise, the smart contract sends the CSP's deposit to the client as compensation. Note that if verification is passed, then the outsourced data are complete, and thus the CSP's deposit is returned. If the auditing verification is not correct, then the outsourced data are incomplete, and the verification result must be incurred by the CSP since the file digest acc_B used for verification is stored on the blockchain, and the blockchain has the property of non-tamperability.

5.2. A Concrete Scheme

The outsourced data are assumed to be static in our scheme. Our hybrid auditing scheme consists of three phases—the set-up phase, the data upload phase, and the audit phase—which extends Ren et al.'s construction [28] as follows:

- Set-up phase: The algorithm of the bilinear pairing instance generator is used to generate cyclic groups G_1, G_2, and G_T with prime order p, a bilinear map e : $G_1 \times G_2 \to G_T$, and s $\overset{R}{\leftarrow} Z_p^*$. For simplicity, we assume $G_1 = G_2 = G$, but this is not essential. The generator of G is denoted by g. Let $sk_{acc} =$ s be the secret key and $pk_{acc} = \left(g, g^s, \cdots, g^{s^n}\right)$ be the public key, where n is the upper bound on the number of elements to accumulate. Each client generates the private key ssk and public key spk for a digital signature algorithm. Then sk_{acc} and ssk are the secret parameters, and the public parameters of our scheme are pk_{acc}, spk, e, and g. Let H be a cryptographic hash function.
- Data upload phase: The client divides the file F into n segments with l_1 bits (i.e., F $= (f_1, \cdots, f_n)$) and then performs the following procedure:
  (1) Each segment is encrypted separately using asymmetric encryption techniques such that $c_i = E(f_i)$ for i $= 1, \cdots,$n.
  (2) An l_2-bit tag τ_i is generated for each segment c_i such that $\tau_i = H(c_i)$ for i $= 1, \cdots,$n. The tags are saved in the tag index table (TIT).
  (3) Each tag is put at the end of its corresponding segment and generates a data block $b_i = c_i \parallel \tau_i$ such that B $= \{b_1, \cdots, b_n\}$.
  (4) The accumulated value of the processed file set B is calculated with the bilinear pair accumulator (i.e., $acc_B = g^{\prod_{i=1}^{n}(b_i+s)}$), and the signature σ $= Sig_{ssk}(name)$ is computed, where the name $\in Z_p$ is the identifier of file B, which is uniformly and randomly chosen by the client.
  (5) The client stores the copies of the TIT, the signature σ, and the auxiliary value aux $= (acc_B, e, g, pk_{acc})$ on the blockchain.
  (6) The processed data file B, the signature σ, and $pk_{acc} = \left(g, g^s, \cdots, g^{s^n}\right)$ are uploaded to the CSP.
- First-phase audit: The client interacts with the CSP as follows:
  (1) The client uses the random index j to challenge the CSP.
  (2) Upon receiving the challenge, the CSP needs to calculate the witness $wit_{b_j} = acc_B^{(b_j+s)^{-1}}$ of element b_j, but the witness cannot be calculated directly since s is unknown.

However, the CSP can express the witness as $wit_{b_j} = \prod_{i=0}^{n-1} \left(g^{s^i}\right)^{a_i}$ using $pk_{acc} = \left(g, g^s, \cdots, g^{s^n}\right)$, where $\{a_0, \cdots, a_{n-1}\}$ is the coefficient of s in polynomial f(s) $= \prod_{b \in B \setminus \{b_j\}} (b+s)$. Note that the CSP uses all elements in B except b_j to compute wit_{b_j}.

(3) The CSP returns (wit_{b_j}, b_j) as a response to the client.

(4) After receiving (wit_{b_j}, b_j), the client checks whether $e(acc_B, g) = e\left(wit_{b_j}, g^{b_j} g^s\right)$ holds. Meanwhile, the client extracts the corresponding segment $c_j{}^*$ and its tag $\tau_j{}^*$ from the block b_j returned by the CSP and compares whether the extracted tag $\tau_j{}^*$ and the original τ_j stored in the TIT are equal. If they are, then $\tau_j' = H\left(c_j{}^*\right)$ is calculated using the extracted data segment $c_j{}^*$,and it is determined whether the calculated tag τ_j' and the original tag τ_j are equal.

If verification is passed, output "1" is determined, meaning the outsourced data are complete; otherwise, output "0" is assigned, which triggers the blockchain's smart contract to perform a second-phase audit for fair arbitration.

- Second-phase audit: The blockchain smart contract interacts with the CSP as follows:

(1) The blockchain smart contract uses the random index j to challenge the CSP.

(2) Upon receiving the challenge, the CSP calculates the witness $wit_{b_j} = \prod_{i=0}^{n-1} \left(g^{s^i}\right)^{a_i}$ of the element b_j by $pk_{acc} = \left(g, g^s, \cdots, g^{s^n}\right)$.

(3) The CSP sends (wit_{b_j}, b_j) as a response to the blockchain smart contract.

(4) After receiving (wit_{b_j}, b_j), the blockchain smart contract checks whether $e(acc_B, g) = e\left(wit_{b_j}, g^{b_j} g^s\right)$ holds. Meanwhile, it extracts the corresponding segment $c_j{}^*$ and its tag $\tau_j{}^*$ from the block b_j returned by the CSP and compares whether the extracted tag $\tau_j{}^*$ and the original τ_j stored in the TIT are equal. If they are, then it calculates $\tau_j' = H\left(c_j{}^*\right)$ using the extracted data segment $c_j{}^*$, and determines whether the calculated tag τ_j' and the original tag τ_j are equal.

If verification is passed, output "1" is reached, meaning the outsourced data are complete, and the smart contract automatically returns the CSP's deposit and submits the deposit of the client to the miner as an audit fee; otherwise, output "0" is reached, meaning the outsourced data are not complete. In this case, the CSP verifies the signature σ by public parameter spk, the smart contract submits the CSP's deposit to the client as compensation only if the outsourced data belong to the client, and then the smart contract submits the deposit of the client to the miners as an audit fee.

In order to better understand the caculations of digest acc_B and the witness in the above audit scheme construction, an illustrative example is given below in Figure 3.

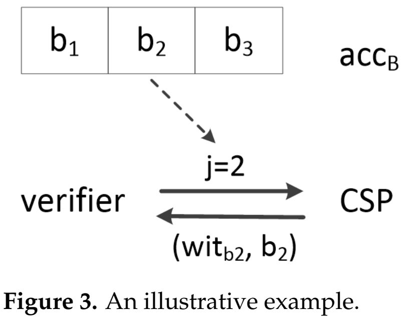

Figure 3. An illustrative example.

Suppose the data processed by the client are B $= \{b_1, b_2, b_3\}$. The client first calculates the digest $acc_B = g^{\prod_{i=1}^{3}(b_i+s)}$. In the audit phase, upon receiving the challenge index j (suppose j = 2) from the verifier (which can be the client or blockchain smart contract), the CSP needs to calculate the witness $wit_{b_2} = g^{\prod_{i=1,i\neq 2}^{3}(b_i+s)}$, but the witness cannot be calculated directly since s is unknown. Let f(s) $= \prod_{i=1,i\neq 2}^{3}(b_i+s) = (b_1+s)(b_3+s) = s^2+(b_1+b_3)s+b_1b_3$. The CSP can express the witness as $wit_{b_2} = \prod_{i=0}^{2}\left(g^{s^i}\right)^{a_i} = (g)^{b_1b_3}\cdot(g^s)^{b_1+b_3}\cdot\left(g^{s^2}\right)^1$ using $pk_{acc} = \left(g, g^s, \cdots, g^{s^n}\right)$, where $\{a_0 = b_1b_3, a_1 = b_1+b_3, a_2 = 1\}$ is the coefficient of s in polynomial f(s). The CSP sends (wit_{b_2}, b_2) as a response to the verifier. We can see that the CSP needs to use all elements in B except b_j to compute the witness wit_{b_j}.

6. Analysis of Our Design

In this section, we analyze the security and characteristics of our scheme. It is assumed that the underlying cryptographic tools such as the bilinear pairing instance generator algorithm, bilinear pair accumulator, one-way hash function, asymmetric encryption algorithm, and digital signature algorithm are secure.

6.1. Security Analysis

We should ensure the correctness and soundness requirements in our scheme. Correctness means that the response provided by the CSP can pass verification if the outsourced data on the CSP are not corrupted. Soundness means that verification can be passed only if the outsourced data on the CSP is not broken. We give the following theorems to prove these requirements can be satisfied in our proposed scheme:

Theorem 1. *For correctness, suppose that both the CSP and client execute the proposed scheme honestly. If the outsourced data on the CSP are not broken, then the CSP's response can pass verification.*

Proof. Suppose that both the CSP and client execute the proposed scheme honestly. As described in our scheme, the CSP's response (wit_{b_j}, b_j) can pass verification only when the following two conditions are met:

(1) $e(acc_B, g) = e\left(wit_{b_j}, g^{b_j}g^s\right)$.

(2) By extracting the data segment ${c_j}^*$ and its corresponding tag ${\tau_j}^*$ from the target block b_j returned by the CSP, the extracted tag ${\tau_j}^*$ and the original τ_j stored in the TIT are equal. By calculating $\tau_j' = H({c_j}^*)$ using the extracted data segment ${c_j}^*$, the calculated tag τ_j' and the original tag τ_j are equal.

If the outsourced data on the CSP are not broken, then the first condition is met since the following equations hold due to the properties of bilinear mapping:

$$\begin{aligned} e(acc_B, g) &= e\left(g^{\prod_{i=1}^{n}(b_i+s)}, g\right) \\ &= e(g,g)^{\prod_{i=1}^{n}(b_i+s)} \\ &= e\left(g^{\prod_{b_i\in B\setminus\{b_j\}}(b_i+s)}, g^{(b_j+s)}\right) \\ &= e\left(wit_{b_j}, g^{b_j}g^s\right) \end{aligned}$$

where for the data file digest $acc_B = g^{\prod_{i=1}^{n}(b_i+s)} = g^{\prod_{b_i\in B}(b_i+s)}$, the witness response by the CSP is $wit_{b_j} = g^{\prod_{b_i\in B\setminus\{b_j\}}(b_i+s)}$.

Meanwhile, the client extracts the corresponding segment ${c_j}^*$ and its tag ${\tau_j}^*$ from the target block b_j returned by the CSP. If the outsourced data are complete, then the extracted tag ${\tau_j}^*$ and the original τ_j stored in the TIT must be equal, and the calculated $\tau_j' = H({c_j}^*)$ using the extracted data segment ${c_j}^*$ must be equal to the original tag τ_j stored in the TIT, so the second condition is met. Therefore, if the outsourced data on the CSP are not broken, then the CSP's response can pass verification. □

Theorem 2. *Regarding soundness, verification is only possible if the outsourced data on the CSP are not corrupted. In other words, if the outsourced data are not complete, then verification cannot be passed.*

Proof. In our proposed scheme, verification can be passed in either the first-phase auditing or the second-phase auditing. In both cases, after receiving (wit_{b_j}, b_j) from the CSP, verification can be passed only when the following two conditions are met:

(1) $\mathrm{e}(acc_B, g) = \mathrm{e}\left(wit_{b_j}, g^{b_j} g^s\right)$.

(2) When extracting the data segment $c_j{}^*$ and its corresponding tag $\tau_j{}^*$ from the target block b_j returned by the CSP, the extracted tag $\tau_j{}^*$ and the original τ_j stored in the TIT are equal, and by calculating $\tau_j' = H(c_j{}^*)$ using the extracted data segment $c_j{}^*$, the calculated tag τ_j' and the original tag τ_j are equal.

Upon receiving the challenge, the CSP needs to compute the corresponding witness wit_{b_j} for the target data block b_j, and all data blocks except b_j must be used in the calculation of the witness. Thus, even a small change in the outsourced data can cause the generated witness to change. Moreover, the CSP returns both the target data block and the calculated witness as a response. If the target data block b_j is corrupted, then either the extracted tag $\tau_j{}^*$ is not equal to the original τ_j stored in the TIT or the calculated $\tau_j' = H(c_j{}^*)$ using the extracted data segment $c_j{}^*$ is not equal to the original tag τ_j stored in the TIT. In other words, due to the security of the hash algorithm, it is almost impossible to generate the same tags using other data blocks. Thus, a valid response cannot be generated by the CSP if the data are not actually saved or the data are corrupted on the CSP. Therefore, if the outsourced data are not complete, then verification cannot be passed. □

6.2. Other Characteristics

Our scheme has the following properties:

- Privacy preservation: Before uploading the data, the client divides the data file F into n segments with l_1 bits; that is, $\mathrm{F} = (f_1, \cdots, f_n)$, and then each segment is encrypted separately using asymmetric encryption techniques such that $c_i = E(f_i)$ for $\mathrm{i} = 1, \cdots, \mathrm{n}$. Note that the client does not disclose the key for the encrypted data to others in the whole auditing process so no one can access the outsourced data except the client him or herself;
- Dynamic operations support: Our proposed scheme also supports the dynamic operations of data such as inserting, deleting, and updating by using the tag index table (TIT) similar to the method in [28], ensuring that after dynamic operations, the auditing scheme remains applicable;
- Timely compensation: As described in our scheme, the blockchain smart contract must be triggered for fair arbitration if the outsourced data are corrupted by the CSP. The client signs a smart contract with the CSP, and both the client and CSP send deposits to the smart contract. First, the smart contract submits the deposit of the client to the miners as an audit fee. Secondly, if verification is passed, it returns the CSP's deposit; otherwise, the deposit of the CSP is sent to the client as compensation via the smart contract. Thus, if the outsourced data are corrupted by the CSP, then verification cannot be passed, so the smart contract submits the deposit to the client as compensation automatically after the CSP verifies the signature σ; that is, the client can obtain compensation from the CSP in a timely manner.

7. Performance Evaluation

We implemented our system prototype of our proposed auditing scheme model via Python code. In this scheme, we used Solidity 0.8.11 to build an Ethereum smart contract and used Go Ethereum (Geth) 1.10.16 as the Ethereum client. The smart contract was deployed to the Ethereum test network.

The overhead of the smart contract comes from the posting parameter and on-chain verification. In this scheme, there is just one parameter named $\mathrm{acc_B}$ which is used for on-chain verification. Figure 4 measures the gas costs of different sizes of files that were used in our auditing scheme. It is shown that the cost of our contract implemented on Ethereum was a constant value. The gas cost was fixed at 4.2216×10^4. The total cost of ether could be calculated by the Ethereum gas rule: gasCost $\times$ gasPrice. The average gas price was about 45 Gwei, and 1 Gwei is 10^{-9} ether. The current exchange rate is 1 ether = USD 2500. As shown in Figure 4, our cost for deploying the auditing scheme model was about USD 4.7493. The results confirm that this was not a huge cost for the client.

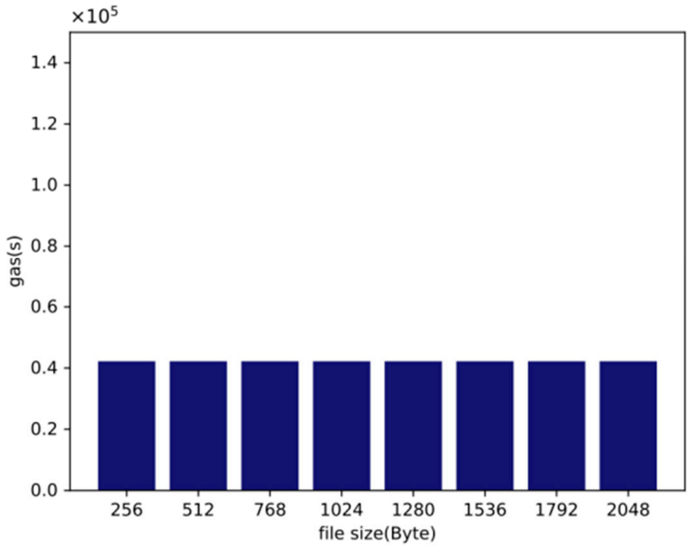

Figure 4. Storage overhead required for blockchain verification based on different file sizes.

In order to execute the polynomial operations in a bilinear map accumulator, we introduced the PBC library in the implementation. The PBC library is an open-source C library built on the GMP library that performs the mathematical operations underlying pairing-based cryptosystems. In the data upload phase, the client encrypts the data blocks which need to be outsourced by RSA, a kind of asymmetric encryption algorithm. Then, the client generates conflict-free 20-bit tags using a hash algorithm. To evaluate the performance of our model, we used the following set-up. The server proxy of the client was collocated with the CSP server on 8 cores of a machine with 2.60 GHz Intel i7-6700 HQ processors and 12 GB of RAM running Ubuntu Linux.

The upload time of the client, the witness computation time of the CSP, and the verification time cost of the verifier should be considered. First, we fixed the size of the data file to 2 MB. The data file was generated randomly from 0–9, a–z, and A–Z in our testing. Both the base size of the data blocks and the size of each increment were designed to be 256 bytes. When the block size was increased to 2 KB, the experiment would be stopped. For each round of experiments, we ran 100 tests with the same data and took the average as the result to reduce the error effect of a single experiment. Note that the larger the data block size, the fewer segments were divided, and the fewer labels were generated since the size of the data file was fixed. The final experimental results are shown in Figure 5a,b. It is not hard to observe that the upload time and the generation time of the witness were inversely proportional to the block size, and the time required for the verifier to conduct verification was fixed at 0.002 s, which conformed to our expectations. We can see that as the block size increased, the time required to calculate acc_B and wit decreased correspondingly, as shown in Figure 5c. Additionally, the order of magnitude of these parameters was about 0.01, well below the upload time. For the client, most of the time the cost was spent on encrypting the data file. Moreover, the calculation overhead of the parameters b (i.e., data segment) and τ (i.e., tag) was independent of the block size.

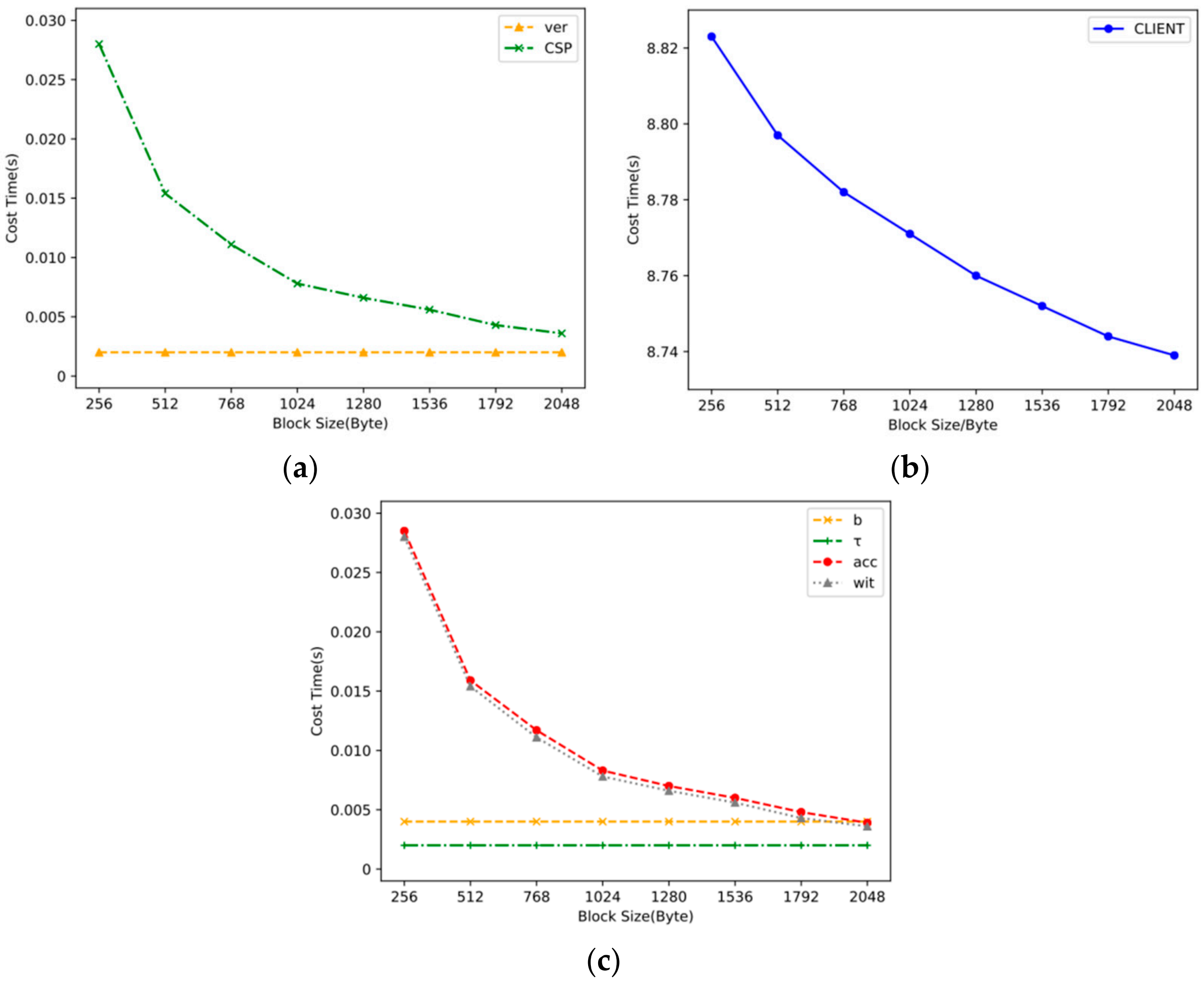

Figure 5. Computing overhead based on different block sizes. (**a**) Computing overhead of audit participants. (**b**) Computing overhead of the client. (**c**) Computing overhead of the parameters.

In addition, if the block size was fixed at 256 bytes, we increased the data file size from 256 KB to 2 MB in increments of 256 KB. Figure 6a,b shows that most of the computing overhead was spent on the data upload phase, and the vast majority of this time was used to encrypt the data as mentioned above. As shown in Figure 6c, both the time required for generating acc_B by the client and the computational overhead for calculating the witness by the CSP were proportional to the size of the outsourced data file. The results show that the computational overhead of the verifier was a constant value regardless of the data file size and the data block size. Therefore, the experimental results show that our proposed auditing scheme was very effective.

To measure the distribution trend of the computing overhead with different parameters, we fixed the size of the data file at 2 MB and the block size at 256 bytes. Then, we repeated the experiment 100 times and recorded the computing overhead of the parameters for each experiment. The distribution trend of the computing overhead is shown in Figure 7. The computing overhead of all parameters was according to Gaussian distribution.

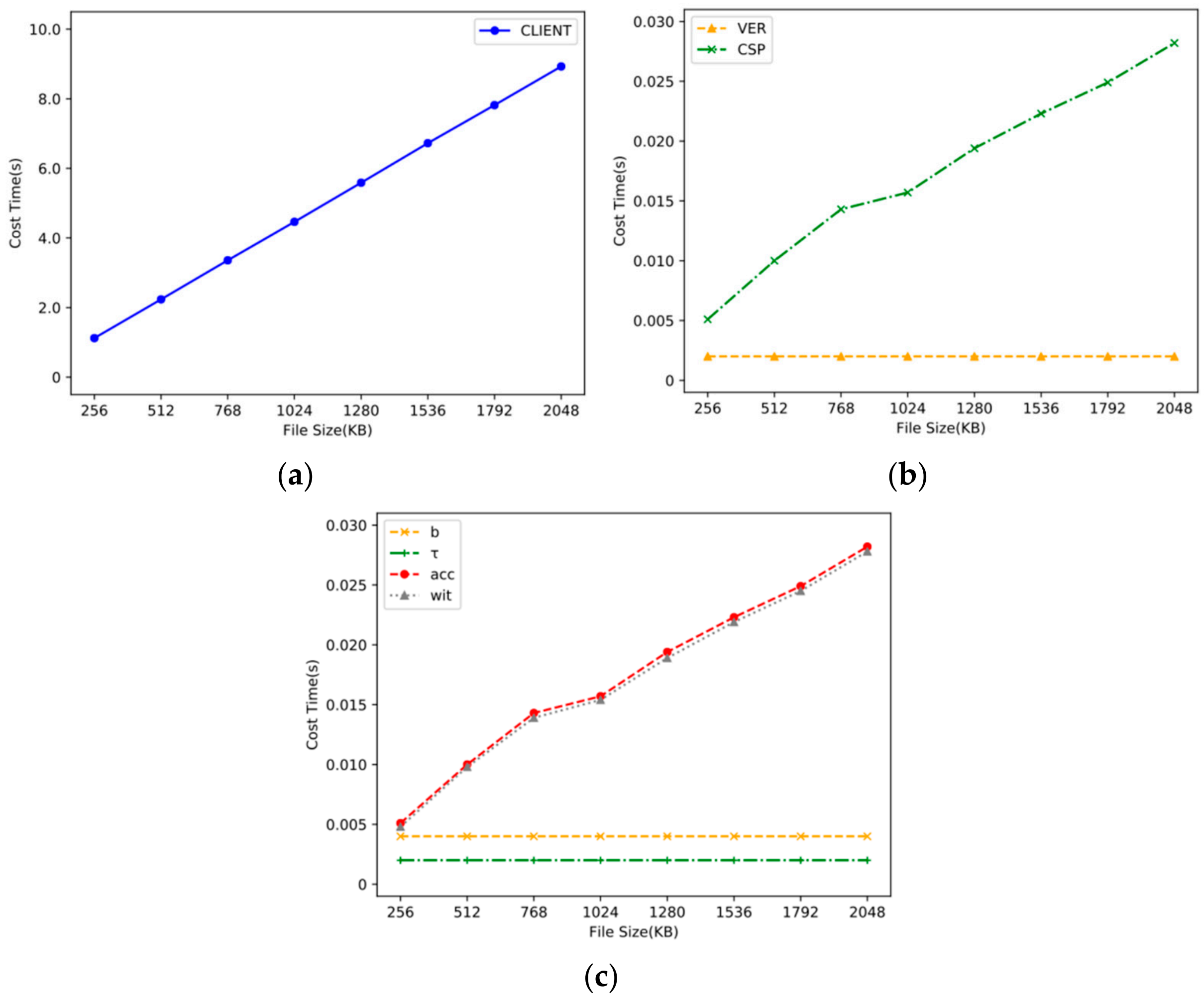

Figure 6. Computing overhead based on different file sizes. (**a**) Computing overhead of audit participants. (**b**) Computing overhead of the client. (**c**) Computing overhead of the parameters.

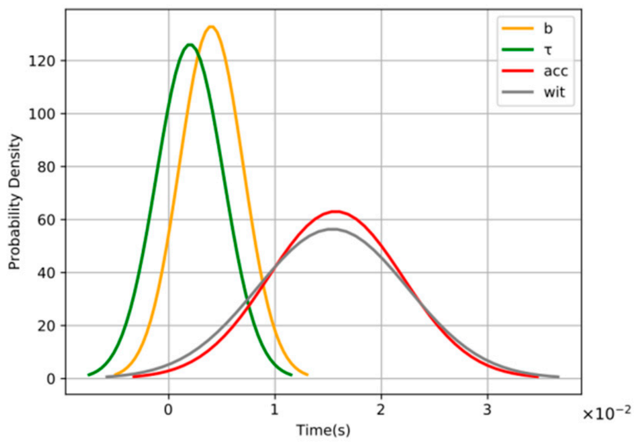

Figure 7. The distribution trend of computing overhead with different parameters.

8. Conclusions and Future Work

In this paper, we designed a novel and efficient two-phase arbitrable hybrid auditing scheme based on the blockchain. By using a bilinear map accumulator and blockchain smart contract, our scheme not only realizes deterministic checking, which provides 100% data possession and integrity guarantees, but also enables a healthy ecosystem between the client and the CSP. That aside, when the outsourced data are lost or corrupted by the CSP, our scheme can compensate the client in a timely manner and punish the dishonest CSP automatically with a smart contract. Meanwhile, our scheme also protects the honest CSP and prevents dishonest behavior from the client. Furthermore, we designed hybrid auditing in our scheme instead of conducting public auditing through the blockchain. The hybrid auditing design not only provides fair judgment but also saves audit fees for the client, because the public auditing phase by the blockchain is triggered only when verification fails in the private auditing phase. Through theoretical and experimental analysis, it was verified that our design was feasible and efficient, and it achieved the desired security goals. Of course, our scheme still has some limitations to be improved upon. For example, it can only check whether the outsourced data are corrupted and cannot determine which data blocks are corrupted or how to repair the corrupted data blocks. In future works, we will enhance more functions of our auditing scheme, such as the location and repair of corrupted data blocks.

Author Contributions: Conceptualization, S.W. and Y.G.; methodology, S.W. and Y.G.; validation, Y.Z. and Y.G.; writing—original draft preparation, S.W.; writing—review and editing, S.W., Y.Z. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities under Grants 2020NTST32 and National Natural Science Foundation of China under Grants 62102035.

Data Availability Statement: The data used to support the findings of this study are available from the authors upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zheng, X.; Cai, Z. A Private and Efficient Mechanism for Data Uploading in Smart Cyber-Physical Systems. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 766–775.
2. Cai, Z.; He, Z. Trading Private Range Counting over Big IoT Data. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 31 October 2019.
3. Zheng, X.; Cai, Z. Privacy-Preserved Data Sharing towards Multiple Parties in Inxdustrial IoTs. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 968–979. [CrossRef]
4. Cai, Z.; Shi, T. Distributed Query Processing in the Edge Assisted IoT Data Monitoring System. *IEEE Internet Things J.* **2021**, *8*, 12679–12693. [CrossRef]
5. Dragulinescu, A.-M.; Constantin, F.; Orza, O.; Bosoc, S.; Streche, R.; Negoita, A.; Osiac, F.; Balaceanu, C.; Suciu, G. Smart Watering System Security Technologies using Blockchain. In Proceedings of the 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 1–3 July 2021.
6. Cai, Z.; He, Z.; Guan, X.; Li, Y. Collective Data-Sanitization for Preventing Sensitive Information Inference Attacks in Social Networks. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 577–590. [CrossRef]
7. Miao, Y.; Ma, J.; Liu, X.; Weng, J.; Li, H.; Li, H. Lightweight fine-grained search over encrypted data in fog computing. *IEEE Trans. Serv. Comput.* **2018**, *12*, 772–785. [CrossRef]
8. Ong, Q.; Miao, Y.; Li, H.; Liu, X.; Deng, R. Privacy-Preserving Ranked Spatial Keyword Query in Mobile Cloud-Assisted Fog Computing. *IEEE Trans. Mob. Comput.* **2021**. [CrossRef]
9. Xie, H.; Guo, Y.; Jia, X. Privacy-preserving Location-based Data Queries in Fog-enhanced Sensor Networks. *IEEE Internet Things J.* **2021**. [CrossRef]
10. Guo, Y.; Xie, H.; Wang, C.; Jia, X. Enabling privacy-preserving geographic range query in fog-enhanced iot services. *IEEE Trans. Dependable Secur. Comput.* **2021**. [CrossRef]
11. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, NY, USA, 2 November 2007–31 October 2007; pp. 598–609.
12. Ateniese, G.; Di Pietro, R.; Mancini, L.V.; Tsudik, G. Scalable and efficient provable data possession. In Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, New York, NY, USA, 22–25 September 2008.
13. Erway, C.C.; Kupcu, A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. *ACM Trans. Inf. Syst. Secur.* **2015**, *17*, 1–29. [CrossRef]
14. Wang, Q.; Wang, C.; Ren, K.; Lou, W.; Li, J. Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. *Parallel Distrib. Syst.* **2011**, *22*, 847–859. [CrossRef]
15. Wang, C.; Chow, S.S.M.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375. [CrossRef]

16. Yuan, J.; Yu, S. Public integrity auditing for dynamic data sharing with multiuser modification. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1717–1726. [CrossRef]
17. Liu, C.; Ranjan, R.; Yang, C.; Zhang, X.; Wang, L.; Chen, J. MuR-DPA: Top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Trans. Comput.* **2014**, *64*, 2609–2622. [CrossRef]
18. Yu, Y.; Li, Y.; Ni, J.; Yang, G.; Mu, Y.; Susilo, W. Comments on 'public integrity auditing for dynamic data sharing with multiuser modification'. *IEEE Trans. Inf. Forensics Secur.* **2015**, *11*, 658–659. [CrossRef]
19. Wang, H.; Qin, H.; Zhao, M.; Wei, X.; Shen, H.; Susilo, W. Blockchain-based fair payment smart contract for public cloud storage auditing. *Inform. Sci.* **2020**, *519*, 348–362. [CrossRef]
20. Yuan, H.; Chen, X.; Wang, J.; Yuan, J.; Yan, H.; Susilo, W. Blockchain-based public auditing and secure deduplication with fair arbitration. *Inf. Sci.* **2020**, *541*, 409–425. [CrossRef]
21. Wang, H.; Li, K.; Ota, K.; Shen, J. Remote data integrity checking and sharing in cloud-based health internet of things. *IEICE Trans. Inf. Syst.* **2016**, *99*, 1966–1973. [CrossRef]
22. Wang, H.; He, D.; Yu, J.; Wang, Z. Incentive and unconditionally anonymous identity-based public provable data possession. *IEEE Trans. Serv. Comput.* **2019**, *12*, 824–835. [CrossRef]
23. Wang, H. Proxy provable data possession in public clouds. *IEEE Trans. Serv. Comput.* **2013**, *6*, 551–559. [CrossRef]
24. Wang, H.; Wang, Q.; He, D. Blockchain-based private provable data possession. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2379–2389. [CrossRef]
25. Wang, H.; He, D.; Fu, A.; Li, Q.; Wang, Q. Provable data possession with outsourced data transfer. *IEEE Trans. Serv. Comput.* **2019**, *14*, 1929–1939. [CrossRef]
26. Kuang, B.; Fu, A.; Yu, S.; Yang, G.; Su, M.; Zhang, Y. ESDRA: An efficient and secure distributed remote attestation scheme for IoT swarms. *IEEE Internet Things J.* **2019**, *6*, 8372–8383. [CrossRef]
27. Zhang, Y.; Yu, J.; Hao, R.; Wang, C.; Ren, K. Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. *IEEE Trans. Dependable Secur. Comput.* **2018**, *17*, 608–619. [CrossRef]
28. Ren, Y.; Qi, J.; Liu, Y.; Wang, J.; Kim, G.-J. Integrity verification mechanism of sensor data based on bilinear map accumulator. *ACM Trans. Internet Technol.* **2021**, *21*, 1–19. [CrossRef]
29. Zafar, F.; Khan, A.; Malik, S.U.R.; Ahmed, M.; Anjum, A.; Khan, M.I.A. 2017. Survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Comput. Security.* **2017**, *65*, 29–49. [CrossRef]
30. Du, Y.; Duan, H.; Zhou, A.; Wang, C.; Au, M.; Wang, Q. Enabling Secure and Efficient Decentralized Storage Auditing with Blockchain. *Proc. IEEE Trans. Dependable Secur. Comput* **2021**. [CrossRef]
31. Caronni, G.; Waldvogel, M. Establishing trust in distributed storage providers. In Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P2003), Linköping, Sweden, 1–3 September 2003; pp. 128–133.
32. Deswarte, Y.; Quisquater, J.-J.; Saïdane, A. *Remote Integrity Checking, in Integrity and Internal Control in Information Systems VI*; Springer: Berlin, Germany, 2004; pp. 1–11.
33. Filho, D.L.G.; Barreto, P.S.L.M. Demonstrating Data Possession and Uncheatable Data Transfer. Cryptology ePrint Archive: Report 2006/150. Available online: https://eprint.iacr.org/2006/150 (accessed on 12 December 2021).
34. Sebe, F.; Domingo-Ferrer, J.; Martinez-balleste, A.; Deswarte, Y.; Quisquater, J. Efficient remote data possession checking in critical information infrastructures. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 1034–1038. [CrossRef]
35. Barsoum, A.F.; Hasan, M.A. *Provable Possession and Replication of Data over Cloud Servers*; University Waterloo: Waterloo, ON, Canada, 2010.
36. 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*, 1432–1437.
37. Buterin, V. Ethereum White Paper. Available online: https://www.mendeley.com (accessed on 12 December 2021).
38. The Storj Project, [Online]. Available online: https://storj.io/storj.pdf (accessed on 12 December 2021).
39. Guo, Y.; Zhang, C.; Jia, X. Verifiable and forward-secure encrypted search using blockchain techniques. In Proceedings of the IEEE International Conference on Communications, Dublin, Ireland, 7–11 June 2020.
40. Guo, Y.; Wang, S.; Huang, J. A blockchain-assisted framework for secure and reliable data sharing in distributed systems. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 1–19. [CrossRef]
41. Tong, W.; Dong, X.; Shen, Y.; Jiang, X. A hierrchical sharding protocol for multi-domain IoT blockchains. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications, Shanghai, China, 20–24 May 2019.
42. Zhu, S.; Cai, Z.; Hu, H.; Li, Y.; Li, W. zkCrowd: A Hybrid Blockchain-based Crowdsourcing Platform. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4196–4205. [CrossRef]
43. Guo, Y.; Xie, H.; Miao, Y.; Wang, C.; Jia, X. FedCrowd: A federated and privacy-preserving crowdsourcing platform on blockchain. In Proceedings of the IEEE Transactions on Services Computing, 14 October 2020.
44. Li, C.; Qu, X.; Guo, Y. TFCrowd: A blockchain-based crowdsourcing framework with enhanced trustworthiness and fairness. *EURASIP J. Wirel. Commun. Netw.* **2021**. [CrossRef]
45. Zhang, C.; Guo, Y.; Jia, X.; Wang, C.; Du, H. Enabling Proxy-Free Privacy-Preserving and Federated Crowdsourcing by Using Blockchain. *IEEE Internet Things J.* **2021**, *8*, 6624–6636. [CrossRef]
46. Wang, M.; Guo, Y.; Zhang, C.; Wang, C.; Huang, H.; Jia, X. MedShare: A Privacy-Preserving Medical Data Sharing System by Using Blockchain. *IEEE Trans. Serv. Comput.* **2021**. [CrossRef]

47. The MedRec Project. Available online: https://www.pubpub.org/pub/medrec (accessed on 28 December 2021).
48. Silvano, F.W.; Marcelino, R. Iota Tangle: A cryptocurrency to communicate Internet-of-Things data—ScienceDirect. *Future Gener. Comput. Syst.* **2020**, *112*, 307–319. [CrossRef]
49. Suciu, G.; Nadrag, C.; Istrate, C.; Vulpe, A.; Ditu, M.-C.; Subea, O. Comparative Analysis of Distributed Ledger Technologies. In Proceedings of the 2018 Global Wireless Summit (GWS), Chiang Rai, Thailand, 25–18 November 2018.