information

MDPI

Article

Public Data Integrity Verification Scheme for Secure Cloud Storage

Yuan Ping [1,*], Yu Zhan [2], Ke Lu [2] and Baocang Wang [2,3,4,*]

1 School of Information Engineering, Xuchang University, Xuchang 461000, China
2 The Sate Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China; yzhan1993@163.com (Y.Z.); 18439925995@163.com (K.L.)
3 Cryptographic Research Center, Xidian University, Xi'an 710071, China
4 School of Information Engineering, Xidian University, Xi'an 710071, China
* Correspondence: pyuan.lhn@xcu.edu.cn (Y.P.); bcwang79@aliyun.com (B.W.)

Received: 11 July 2020; Accepted: 21 August 2020; Published: 25 August 2020

Abstract: Although cloud storage provides convenient data outsourcing services, an untrusted cloud server frequently threatens the integrity and security of the outsourced data. Therefore, it is extremely urgent to design security schemes allowing the users to check the integrity of data with acceptable computational and communication overheads. In this paper, we first propose a public data integrity verification scheme based on the algebraic signature and elliptic curve cryptography. This scheme not only allows the third party authority deputize for users to verify the outsourced data integrity, but also resists malicious attacks such as replay attacks, replacing attack and forgery attacks. Data privacy is guaranteed by symmetric encryption. Furthermore, we construct a novel data structure named divide and conquer hash list, which can efficiently perform data updating operations, such as deletion, insertion, and modification. Compared with the relevant schemes in the literature, security analysis and performance evaluations show that the proposed scheme gains some advantages in integrity verification and dynamic updating.

Keywords: public verification; dynamic data updating; cloud storage; algebraic signatures; elliptic curve discrete logarithm problem

1. Introduction

As a new computing paradigm, cloud storage has been popular for providing great data storage and management services. More and more organizations and individuals outsource their data in the cloud via a pay-as-you-go approach. Nevertheless, cloud outsourcing storage service heavily relieves the local storage burden for the cloud clients. The data integrity verification becomes a critical security challenge for cloud technology to be adopted [1,2]. On the one hand, downloading the complete data frequently for integrity verification will dramatically increase communication and computation overhead. On the other hand, when the cloud storage devices are damaged or hackers steal the outsourced data, the cloud service provider (CSP) might choose to conceal the data corruption or loss for maintaining the user's trust. Furthermore, the CSP might save storage space by intentionally deleting the less frequently accessed data [3] or leak users' privacy information for interest. Hence, cloud users must find an efficient way to verify the integrity of the outsourced data.

The schemes presented in the literature provide models supporting private verification and public verification to solve the above problem. Private verification only allows users to check the outsourced data's integrity by themselves that brings great computing burden to resource constraint users. On the contrary, public verification reduces the user's computational cost by introducing a third-party authority (TPA) to verify integrity. Due to the potential of practical use, public verification

has received more attention [4–11]. Ateniese et al. [4] proposed the first publicly verifiable cloud data integrity scheme, which utilizes the RSA-based homomorphic linear authenticator to check the integrity of outsourced data. Shacham et al. [12] presented a public data integrity verification scheme with the Boneh–Lynn–Shacham (BLS) signature [13]. In this scheme, multiple tags are aggregated into a single tag and verified the tag with bilinear maps. However, these schemes incur heavy communication and computational costs for expensive exponentiation operations. Chen [14] presented an algebraic signatures-based public verification scheme, which used a short bit string compressed by a data block to achieve efficient integrity verification without the comparison with original data. Unfortunately, it is insecure against the replay attack. Furthermore, it fails to support dynamic data updating.

To support dynamic updating, researchers have presented many schemes to update data without downloading the whole data. Erway et al. [5] proposed the first verification scheme, which supports dynamic data updating with a rank-based authenticated skip list. However, the list requires massive auxiliary information to achieve integrity verification. It leads to high communication and computational costs. Subsequently, a number of improved schemes are proposed, such as [3,10,15–18]. Note that a Divide and Conquer Table (DCT) data structure was introduced by [17,18] to support dynamic data updating. According to the table's division into separate sections, DCT performs better than the Index-Hash-Table (IHT) [10] in data block insertion and deletion. However, some drawbacks are still found. First, many continuous storage spaces are essential for the outsourced data blocks that are not practical for some distributed storage. Second, operations such as insertion and deletion result in additional computational costs because data block movement can not be generally avoided. Third, due to the lack of links among the DCTs, it is hard to locate the required data blocks for use quickly. Furthermore, the scheme is fragile to resist the forgery attack, replay attack, and replacing attack.

Considering the above problems, in this paper, we propose a public verification scheme based on the algebraic signature and elliptic curve cryptography. The scheme supports efficient public verification while resisting the possible attacks such as the forgery attack, the replacing attack, and the replay attack. Furthermore, a novel data structure called Divide and Conquer Hash List (DCHL) is introduced to achieve dynamic data updating. DCHL takes advantage of the hash table and linked list to reduce the computational costs. In addition, either in the cloud or during the verification process, data encryption ensures data privacy. Main contributions of the proposed scheme are listed as follows:

- Based on algebraic signature and elliptic curve cryptography, we propose a public verification scheme that supports efficient data integrity verification with low communication and computational overheads. Furthermore, symmetric encryption in the scheme guarantees the privacy of data blocks.
- To support dynamic updating, a novel data structure named DCHL is designed and stored in the TPA to make operations such as data insertion, modification and deletion more flexible and efficient.
- Using the proposed scheme in cloud storage, security analysis suggests that even a malicious CSP cannot launch a forgery attack, replacing attack and replay attack to pass integrity verification. Meanwhile, the proposed scheme frequently outperforms the relevant data verification schemes in terms of efficiency that is confirmed by numerical analysis and real experiments.

The remainder of the paper is organized as follows. Section 2 briefly reviews the related works. The system model and the design goals are described in Section 3. In Section 4, some necessary preliminaries are presented for a better understanding of the proposed scheme. Section 5 presents the details of the scheme and the corresponding dynamic updating operations. Section 6 gives the security analysis of our scheme, while its comparisons with the state-of-the-art schemes are shown in Section 7. Finally, conclusions are drawn in Section 8.

2. Related Work

Recently, growing attention has been paid to data integrity verification in the cloud storage. In 2007, Juels et al. [19] proposed the concept of "proof of retrievability (PoRs)". This scheme not only verifies the integrity of data stored in the cloud, but also ensures data retrievability with error-correcting code. Nevertheless, it only can audit privately [15]. In 2008, Ateniese et al. [20] first proposed a scheme that could verify the integrity of data publicly in the cloud storage. Their scheme combines the block tags into a single one with the RSA-based Homomorphic Verifiable Tag (HVT) without downloading all the data. However, researches [17,18,21,22] indicated that this technique uses lots of communication and computational resources since it utilized the RSA numbers. Shacham et al. [12] presented a public data integrity verification scheme with the BLS signature. Since the model operation is not required to be carried out, the BLS [13] consumes less communication and computation resources. However, a malicious attacker could access the outsourced data in interactive proof systems that threaten data privacy.

In 2013, Wang et al. [6] presented a privacy-preserving auditing scheme. It combines random mask with the homomorphic to prevent the TPA from getting any information about the outsourced data in the cloud. Later, Worku et al. [23] proposed a privacy-preserving public auditing scheme of which performance is considered better than [6]. It verifies data integrity with a ring signature and protects the identity information of users. Recently, Yu et al. [24] proposed an identity-based verification scheme to protect data privacy, which uses the identity information of the user as a key to verify the data integrity. However, these privacy protection schemes are extremely unsafe for the outsourced data. Furthermore, the above schemes incur high communication and computational costs.

Based on an algebraic signature, Chen et al. [14] proposed an efficient integrity verification scheme. It improves the verification efficiency without using the public key technique. However, this scheme achieves an unlimited number of verifications by frequently updating the challenge process, which causes extra communication and computational costs. Furthermore, this scheme is vulnerable to a replay attack. Recently, Sookhak et al. [17,18] presented a scheme supporting unlimited verification. However, some security vulnerabilities are also found. The CSP is likely to compute the secret key of the signature with the remaining data blocks and tags when some data blocks and tags are corrupted. In addition, this scheme cannot resist replay attack and replacing attack.

Generally, we cannot guarantee that the outsourced data is kept unchanged during the whole period of the cloud [25–29]. Users might update their data and require the integrity of updated data to be verified. More and more researches have put forward many schemes to update data without downloading the whole data efficiently. For instance, Erway et al. [5] utilized a rank-based authenticated skip list to construct a dynamic provable data possession scheme. The scheme supports dynamic updating by adding the rank information at an RSA tree. Wang et al. [3] proposed a typical public auditing scheme with a Merkle Hash Tree to achieve dynamic updating. When we want to update data, the CSP updates the leaf node of the MHT and re-calculates the root of the new MHT. However, these schemes are frequently inapplicable due to pricey computation and communication resources required by their updating procedures.

Zhu et al. [10] designed the IHT to support dynamic updating, for example, fast data modification. However, additional computational costs for data insertion and data deletion are essential. Schemes in [15,16] utilized a Dynamic Hash Table (DHT) and a Doubly Linked Info Table (DLIT) to achieve dynamic updating operations. They take advantage of the linked list to reduce the cost of data insertion and deletion. However, along with the stored list increases, the query time would increase dramatically in the verification and update. Then an emerging increase in the computational cost cannot be avoided.

Later, Sookhak et al. [17,18] utilized the DCT to achieve dynamic updating. They split a large table into several separate parts. Although it seems that the costs of data insertion and deletion are reduced, new problems occur. For instance, multiple contiguous blocks of memory are required,

and the connection between data is also interrupted by the fragmentation of the table that undoubtedly increases data access time.

3. System Model and Design Goals

Initially, we illustrate the system model and the design goal of the proposed scheme.

3.1. System Model

As shown in Figure 1, the proposed public verification scheme adopts a three-party model. Roles in this model are as follows: (1) Users, who draw support from a cloud to store their data. (2) Cloud Server Provider (CSP), which sells abundant storage and computational resources to users. (3) Third-Party Authority (TPA), who verifies the integrity of data in response to user requests.

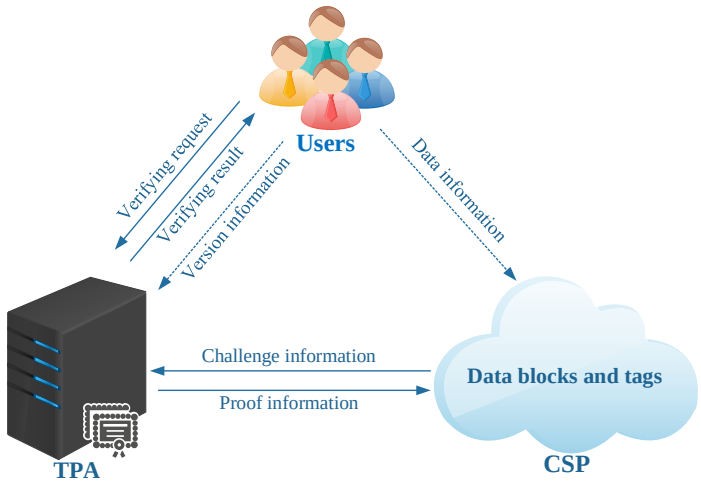

Figure 1. The system model for the proposed scheme.

Following [12], we present the ability of each party for the proposed model. First, the TPA is considered to be honest-but-curious. In other words, the TPA is honest in verifying data integrity, but it might be curious about the data. Furthermore, the CSP is untrusted because it can choose to conceal the data corruption or loss for maintaining the user's trust. Therefore, the CSP can launch the following attacks.

- Forgery attack: The CSP might forgery-proof information to pass verification during which the outsourced data is deleted or corrupted.
- Replacing attack: The CSP might replace the corrupted data blocks and tags with other valid uncorrupted data blocks and tags if the challenged data blocks are corrupted.
- Replay attack: To pass verification, the CSP might send the former valid proof information or other information to the TPA.

3.2. Design Goals

To guarantee security and efficiency, the design goals should achieve the following requirements:

- *Public verification*: The scheme allows the TPA to verify the outsourced data's integrity as an agent.
- *Correctness*: If the CSP correctly stores the user's data, it could successfully pass integrity verification.
- *Privacy preserving*: The scheme can be securely stored in the cloud and prevent privacy from leaking during the verification process.
- *Unforgeability*: If the outsourced data is corrupted, the CSP cannot forge the proof information to deceive the TPA.

- *Dynamic data updating*: Users could perform the modification, insertion, and deletion operation on the data stored in the cloud.
- *Lightweight*: The scheme requires low communication and computational costs in verification and dynamic updating.

4. Preliminaries

4.1. Elliptic Curve Discrete Logarithm Problem

Let $E\left(F_q\right)$ be an elliptic curve over a large prime q order finite field F_q, where q is a large prime. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is a well-known hard problem, which can be described as follows: Given an equation $A = xB$, where $A, B \in E\left(F_q\right)$ and $x < q$. It is easy to compute A with x and B. However, it is hard to compute x with A and B. Here, we omit the detailed proof, which can be found in [30].

4.2. Algebraic Signatures

According to Litwin et al.'s definition [31], the algebraic signature defined in the Galois field is a type of hash function and supports the homomorphic addition. It is utilized to calculate data tags and verify the integrity of the outsourced data f including n data blocks $\{m_1, \cdots, m_i, \cdots, m_n\}$. Therefore, the algebraic signature of data f can be computed by

$$Sig_\alpha\left(f\right) = \sum_{i=1}^{n} m_i \alpha^i. \tag{1}$$

Here $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ is a vector of different non-zero elements $\alpha_i (i \in [1, n])$ of the Galois field.

An algebraic signature could ensure that the signature for the sum of some random blocks is equal to the sum of the signatures for these blocks [17,18,31]. The property can be described by

$$Sig_\alpha\left(m_1 + \cdots + m_n\right) = Sig_\alpha\left(m_1\right) + \cdots Sig_\alpha\left(m_n\right), \tag{2}$$

where $(m_1, \cdots, m_n)$ are n data blocks.

When all the data blocks are identical, a property of the algebraic signature can be obtained through

$$Sig_\alpha\left(nm\right) = nSig_\alpha\left(m\right). \tag{3}$$

5. The Proposed Scheme

5.1. Divide and Conquer Hash List

To support dynamic data updating, in this section, a novel data structure DCHL is designed. It combines the advantages of the hash table and linked list stored in the TPA. Differing from the DCT, the DCHL is composed of a hash table and many linked lists. The user divides the data blocks into k groups and each group's length is L_i $(i = 1, \cdots, k)$. The index and length of each group are stored in the hash table. The data version information (VI) in the same group is linked with pointers. Here, VI is a number indicating the version of the data block. For each VI, the storage location is unique in the DCHL. When we query the VI of a data block, the TPA quickly locates the group location of VI in the hash table and then traverses the linked list to get the precise location based on the length of the linked list. Figure 2 details a sample structure of the DCHL. Based on Figure 2, for instance, when the TPA queries the ith data block, the TPA first locates the group index of the ith data block and gets the determined group's length. Then, the TPA finds the corresponding position on the determined linked list. Finally, the queried VI will be found.

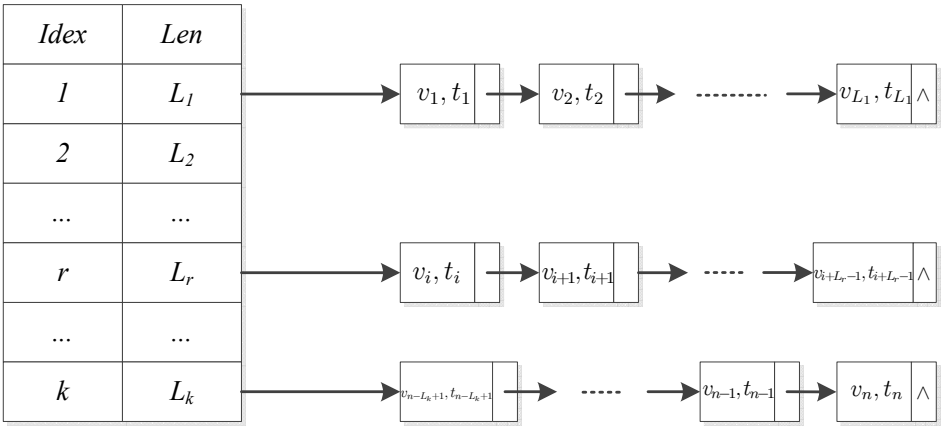

Figure 2. Divide and conquer hash list.

This data structure has advantages over the DCT in schemes [17,18]. Firstly, the TPA only needs to modify the list pointer for insertion and deletion operations. Secondly, only a continuous space is required to store and manage the group index, which is very acceptable in practice. Thirdly, the TPA can flexibly adjust the DCHL if the insertion or deletion operations are frequent. Thus, the data structure can significantly reduce the computational and communication costs of updating processes. A simulated experiment in the following text will demonstrate the conclusion.

5.2. *Verification Scheme Against Malicious Attacks*

Let F_q be a large prime q order finite field, and $E(F_q)$ be an elliptic curve defined in the F_q. Set the point G as the base point of $E(F_q)$. H is a collusion-resistance hash function $H: \{0,1\}^* \rightarrow \{0,1\}^l$. Assume that the outsourced data is divided into n data blocks, i.e., $f = \{m_1, \cdots, m_i, \cdots, m_n\}$ and the length of each data block is l. The proposed verification scheme against malicious attacks involves the following steps:

(1) ***Key initiation***: The user first generates a symmetric key dk for encrypting data blocks. Then, he randomly selects $x \in \mathbb{Z}_q$, and calculates $G_A = xG$, where G is known by the user and the TPA. Meanwhile, the user chooses a secure element α for an algebraic signature. Here, we set (dk, x) as the secret key and G_A as the public key.

(2) ***Data blocks encryption***: The user uses a symmetric encryption algorithm $Enc(\cdot)$ with key dk to encrypt each data block m_i and get the encrypted $M_i = Enc_{dk}(m_i)$ where $1 \leq i \leq n$.

(3) ***Tag initiation***: The user computes the data block tag σ_i for each encrypted data block M_i

$$\sigma_i = Sig_\alpha \left(x \left(M_i + H \left(v_i \| t_i \right) \right) \right), \tag{4}$$

where v_i is the version of the data block M_i, t_i is the timestamp, and $||$ denotes concatenation. Then, the user uploads $\{f, \{\sigma_i, 1 \leq i \leq n\}\}$ to the CSP, and routes $\{v_i, t_i, 1 \leq i \leq n\}$ as the VI to the TPA. After that, the user removes the locally stored information. The TPA is delegated to launch a verification challenge to verify the integrity of the outsourced data. Notice that the proposed scheme respectively uses dk to protect data block and x in tag initiation for public verification. By exploiting the property of ECDLP, it is hard for attackers to extract x with $x(M_i + H(v_i \| t_i))$ and the public key G_A.

(4) ***Challenge***: First, the user transmits a verification request to the TPA. Then, the TPA randomly chooses c data blocks from n data blocks. Finally, the TPA launches a challenge by sending the challenge information $chal = \{s_1, \cdots, s_c\}$ to the CSP where $s_i (1 \leq i \leq c)$ is the index of the selected data block.

(5) ***Proof generation***: After receiving the challenge information, the CSP first calculates $M' = \sum_{i \in chal} M_i$ and $\sigma = \sum_{i \in chal} \sigma_i$, where M' is the data proof and σ is the tag proof. Then, the CSP returns (M', σ) to TPA as the proof.

(6) ***Proof verification***: The TPA calculates the sum of hash values by

$$H' = \sum_{i \in chal} H(v_i \| t_i). \tag{5}$$

Then, the TPA checks the following equation to verify the proof:

$$G_A \cdot \left(Sig_\alpha \left(M' + H'\right)\right) \stackrel{?}{=} G \cdot \sigma \tag{6}$$

Equation (6) outputs 'TRUE' if the equation holds. Finally, the TPA sends the result to the user. Figure 3 illustrates the challenge-verification procedure.

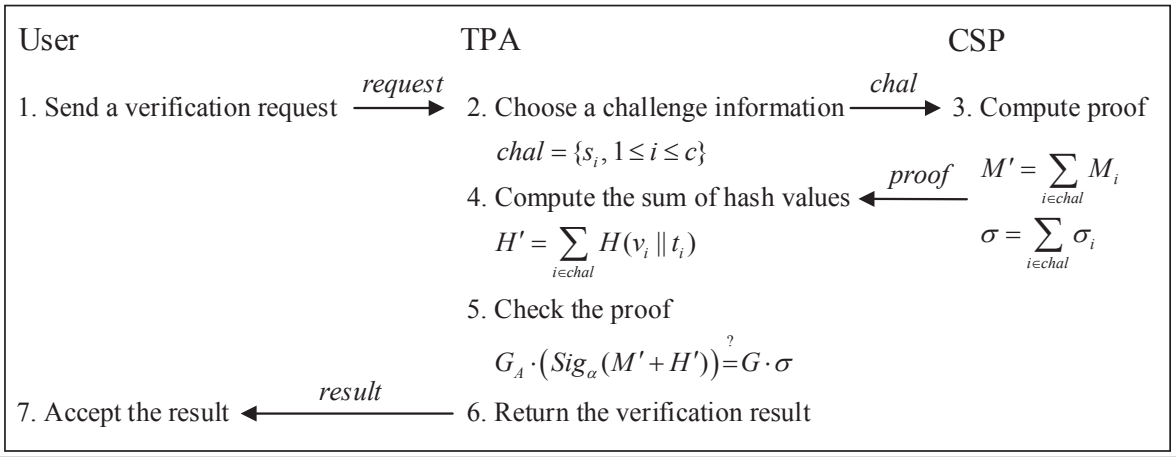

Figure 3. The challenge-verification procedure.

5.3. Dynamic Data Updating

Dynamic data updating operations mean that the users could update their outsourced data without downloading the whole data. These operations include data modification (U_M), data insertion (U_I) and data deletion (U_D). In practice, it is unreasonable for the user to download the entire data before he can update some data blocks. So, we utilize the DCHL to support efficient data updating operations.

5.3.1. Data Modification

Data modification operation replaces a number of blocks with new blocks. Given a new block M_i', the modification algorithm is performed by the following six steps:

(1) With the help of TPA, the user finds the specific DCHL that has the required block and gets the version number v_i. Then, the user generates the new version and timestamp $\left(v_i' \leftarrow v_i + 1, t_i'\right)$ for M_i', and then calculates the tag of the data block M_i' by

$$\sigma_i' = Sig_\alpha \left(x \left(M_i' + H\left(v_i' \| t_i'\right)\right)\right). \tag{7}$$

(2) The user sends the data updating request $U_{M_{CSP}} = \left\{i, M_i', \sigma_i'\right\}$ and the VI updating request $U_{M_{TPA}} = \left\{i, v_i', t_i'\right\}$ to the CSP and the TPA, respectively.

(3) After receiving $U_{M_{CSP}} = \left\{i, M_i', \sigma_i'\right\}$, the CSP replaces the block with M_i' and changes tag σ_i to σ_i'.

(4) After receiving $U_{M_{TPA}} = \{i, v_i', t_i'\}$, the TPA first finds the group index of the i-th data block from the DCHL;
(5) Then, the TPA determines the location of the data block, which needs to be modified in the linked list.
(6) Finally, the TPA modifies $\{v_i, t_i\}$ to $\{v_i', t_i'\}$.

Figure 4 describes the modification operation on DCHL.

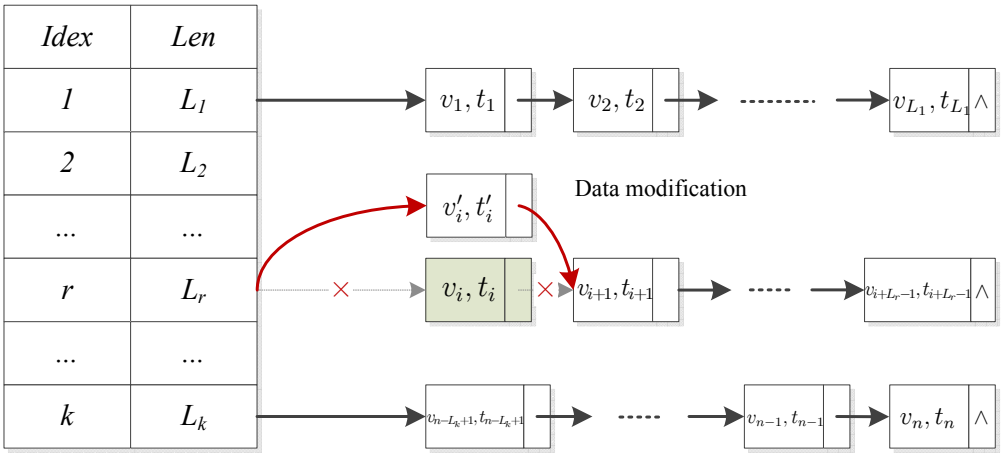

Figure 4. Data block modification operation.

5.3.2. Data Insertion

If the entire file is not required to be downloaded, the user can insert a new data block after some particular positions. Suppose the user wants to insert a new block after M_i, he has to perform as follows:

(1) The user first generates the version and timestamp $\left(v_i^*, t_i^*\right)$ for M_i^*, and then he computes the tag by

$$\sigma_i^* = Sig_\alpha \left(x \left(M_i^* + H \left(v_i^* \| t_i^*\right)\right)\right). \tag{8}$$

(2) The user respectively sends the data updating request $U_{I_{CSP}} = \{i, M_i^*, \sigma_i^*\}$ and the VI updating request $U_{I_{TPA}} = \{i, v_i^*, t_i^*\}$ to the CSP and the TPA.
(3) The CSP inserts data block $M_i{}^*$ after M_i and stores the corresponding tag σ_i^* upon receiving $U_{I_{CSP}} = \{i, M_i^*, \sigma_i^*\}$.
(4) After receiving $U_{I_{TPA}} = \{i, v_i^*, t_i^*\}$, the TPA finds the position of i-th data block in the DCHL and inserts $\{i, v_i^*, t_i^*\}$ after it. Finally, the TPA sets the length of the group in which the i-th data block is located $L_r = L_r + 1$.

Figure 5 depicts the insertion operation in the DCHL.

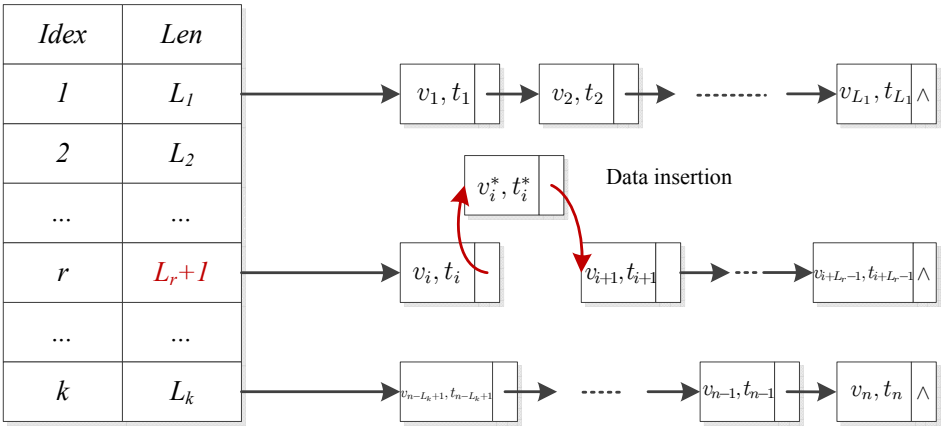

Figure 5. Data block insertion operation.

5.3.3. Data Deletion

Data deletion operation deletes a specific data block. If a user wants to delete the data block M_i, he achieves this objective according to the following steps.

(1) The user sends the data updating request $U_{D_{CSP}} = \{i\}$ and the VI updating request $U_{D_{TPA}} = \{i\}$ to the CSP and the TPA, respectively;
(2) After receiving $U_{D_{CSP}} = \{i\}$, the CSP deletes the block M_i and tag σ_i, respectively;
(3) Similar to the insertion operation, the TPA deletes (v_i, t_i) and sets the length of the group in which the i-th data block is located $L_r = L_r - 1$ in DCHL.

Figure 6 illustrates this deletion operation.

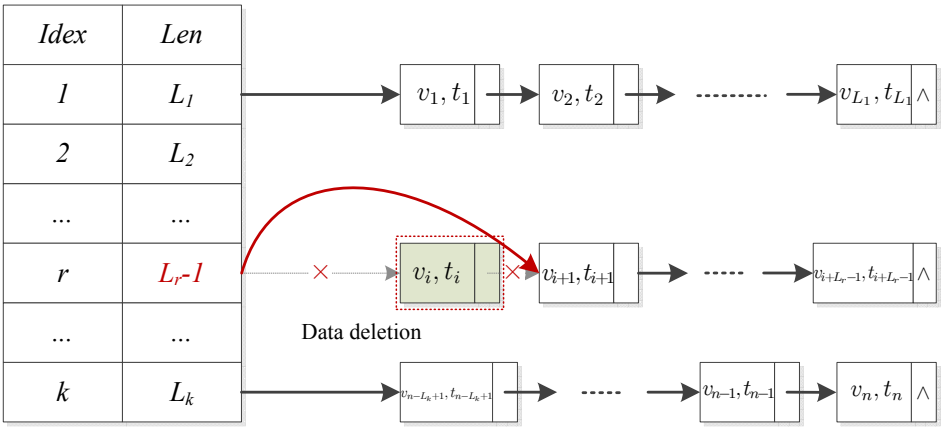

Figure 6. Data block deletion operation.

6. Security Analysis

To conduct security analysis, in this section, the completeness of the proposed scheme is first elaborated. Then, we prove that the scheme resists some attacks well.

Theorem 1 (Completeness)**.** *The CSP pass the verification if it correctly stores the outsourced data according to the TPA's challenge.*

Proof. Given a response information (M', σ) from the CSP, where $M' = \sum_{i \in chal} M_i$, the TPA computes $H' = \sum_{i \in chal} H(v_i \| t_i)$. Then, the TPA gets

$$G_A \cdot \left(Sig_\alpha \left(M' + H'\right)\right) = G \cdot x \cdot \left(Sig_\alpha \left(M' + H'\right)\right) \quad (9)$$

According to Equation (3), the TPA computes

$$\begin{aligned} G \cdot x \cdot \left(Sig_{\alpha}\left(M' + H'\right)\right) &= G \cdot Sig_{\alpha}\left(x \cdot \left(M' + H'\right)\right) \\ &= G \cdot \sum_{i \in chal} Sig_{\alpha}\left(x \cdot \left(M_i + H_i\right)\right) \\ &= G \cdot \sigma \end{aligned} \tag{10}$$

Thus, the equation holds if and only if the CSP correctly stores the outsourced data according to the TPA's challenge. This completes the proof of the theorem. □

Lemma 1 (Unforgeability of the proof)**.** *The CSP cannot pass the verification by forging a proof.*

Proof. If a data block or tag is corrupted, the CSP might falsify a tag proof and a data proof to pass the verification. Following [12], we define the security game as follows.

According to the challenge information $chal = \{s_i, 1 \leq i \leq c\}$, the CSP generates a forgeable proof (M'', σ'). If it cannot pass the verification, the CSP loses the game; otherwise, the TPA has

$$G_A \cdot [Sig_{\alpha}(M'' + H')] = G \cdot \sigma'. \tag{11}$$

Then, from Equation (2), the TPA can learn that

$$G_A \cdot Sig_{\alpha}\left(M''\right) + G_A \cdot Sig_{\alpha}\left(H'\right) = G \cdot \sigma'. \tag{12}$$

According to Equation (3), Equation (12) can be reformulated by

$$G \cdot \sigma' - G_A \cdot Sig_{\alpha}\left(M''\right) = G_A \cdot Sig_{\alpha}\left(H'\right). \tag{13}$$

Then, the TPA gets

$$G \cdot \left(\sigma' - x \cdot Sig_{\alpha}\left(M''\right)\right) = G \cdot \left(x \cdot Sig_{\alpha}\left(H'\right)\right). \tag{14}$$

Let P be $G \cdot \left(x \cdot Sig_{\alpha}\left(H'\right)\right)$ and k denote $\sigma' - x \cdot Sig_{\alpha}\left(M''\right)$, Equation (14) can be rewritten by

$$G \cdot k = P. \tag{15}$$

Apparently, the CSP can pass the verification if and only if there is an efficient algorithm to solve the ECDLP. Unfortunately, this is in contradiction with the assumption. Thus, we can conclude that the CSP cannot forge the proof to pass the integrity verification successfully. □

Based on the aforementioned analysis, we can further prove that the proposed scheme is immunity from replacing attack and replay attack.

Lemma 2 (Immunity from replacing attack)**.** *For verification, it is unavailing for the CSP to replace a specified block and the corresponding tag with others.*

Proof. Similarly, the replacing attack game can be described as follows. The TPA sets $chal = \{s_i, 1 \leq i \leq c\}$ as the challenge information and transmits it to the CSP. After receiving *chal*, the CSP returns a checking proof (M'', σ'). When the proof is generated, the information of the j-th block is replaced with the k-th block's $(k \neq j)$. The CSP will win the game if the proof is verified.

If the CSP wins the game, the TPA thus gets

$$G_A \cdot Sig_{\alpha}(M'' + H') = G \cdot \sigma'. \tag{16}$$

It can deduce that

$$G_A \cdot Sig_\alpha(M'' + H') = G_A \cdot (\sum_{\substack{i \in chal \\ \& i \neq j}} M_i + M_k + \sum_{\substack{i \in chal \\ \& i \neq j}} H(v_i \| t_i) + H(v_j \| t_j)). \tag{17}$$

The right half of Equation (17) can be computed by

$$G \cdot \sigma' = G \cdot (\sum_{\substack{i \in chal \\ \& i \neq j}} \sigma_i + \sigma_k). \tag{18}$$

The TPA thus get

$$G_A \cdot [Sig_\alpha(M_k + H(v_j \| t_j))] = G \cdot \sigma_k. \tag{19}$$

On the basis of the properties of the algebraic signature, Equation (19) can be expressed as

$$\begin{aligned} G_A \cdot Sig_\alpha(M_k + H(v_j \| t_j)) &= G \cdot Sig_\alpha(x(M_k + H(v_k \| t_k)) \\ &= G_A \cdot Sig_\alpha(M_k + H(v_k \| t_k)). \end{aligned} \tag{20}$$

Thus, the TPA obtains

$$Sig_\alpha(M_k + H(v_j \| t_j)) = Sig_\alpha(M_k + H(v_k \| t_k)). \tag{21}$$

If Equation (21) holds, the equation $H(v_j \| t_j) = H(v_k \| t_k)$ holds either. It means that $v_j = v_k$ and $t_j = t_k$. However, we have $t_j \neq t_k$ since $j \neq k$. In addition, the hash function is collision resistant. Hence, $H(v_j \| t_j)$ is not the same as $H(v_k \| t_k)$. Consequently, the CSP loses the game, and the proposed is immune from the replacing attack. □

Lemma 3 (Immunity from replay attack)**.** *For the proposed scheme, the CSP is incapable of passing the verification with the former challenge information.*

Proof. First, the TPA transmits the challenge information to the CSP. After receiving the challenge information, the CSP returns the checking proof (M'', σ'). Similarly, the information of the j-th block is replaced by its previous information while the proof is generated. The CSP will win the game if and only if it can pass the verification. Unfortunately, the CSP cannot pass the verification since the current timestamp is different from the old one. The details are omitted here. □

Theorem 2 (Security)**.** *The proposed scheme is secure against the threat model.*

Proof. As discussed in Section 3.1, the threat model of the proposed scheme consists of forgery attack, replacing attack, and replay attack. From Lemmas 1, 2, and 3, the CSP cannot win in the three attacks. Thus, we can conclude that the proposed scheme is secure against the threat model. □

7. Performance Analysis

7.1. Misbehavior Detection

Towards an affordable and practical verification with high-accuracy for both the TPA and the CSP, sampling verification is preferred by our following scheme [15,17,18]. If the outsourced data is divided into n blocks, and m out of n are corrupted (or modified by the CSP), we can check random sampled c blocks to detect the misbehavior. Generally, the fractions of the corrupted data is $\frac{m}{n}$. Therefore, the detection probability of verification is $P = 1 - (1 - m/n)^c$. Particularly, when $m/n = 0.02$, the probability of detection is greater than 0.9976 if the TPA verifies 300 randomly chosen blocks.

7.2. Communication Costs

In this section, we discuss the communication costs compared with the state-of-the-art schemes. The challenge-verification process mainly causes communication costs. First, the user consigns the TPA to verify the integrity of the outsourced data with $O(1)$ communication costs. The TPA launches a challenge with $O(c)$ communication costs, where c is the number of challenging blocks. Then the CSP sends the proof to the TPA, and this step brings $O(1)$ communication costs. Finally, the TPA transmits the verifying result to the user, bringing $O(1)$ communication costs. Hence, the communication costs of the whole process are $O(c)$. Table 1 shows the communication costs compared with several popular auditing schemes, such as Erway et al.'s scheme [5], Chen et al.'s scheme [14], Tian et al.'s scheme [15], Shen et al.'s scheme [16] and Sookhak et al.'s [17,18]. We can find that the communication costs are the same as the latter two schemes but more effective than the former.

Table 1. Comparison of communication costs.

Schemes	Communication Costs
Erway [5]	$O(c \log n)$
Chen [14]	$O(c)$
Tian [15]	$O(c)$
Shen [16]	$O(c)$
Sookhak [17,18]	$O(c)$
Our scheme	$O(c)$

7.3. Computational Costs

In terms of computation complexity, we compare our scheme with the state-of-the-art schemes, i.e., Chen et al.'s scheme [14], Tian et al.'s scheme [15], Shen et al.'s scheme [16] and Sookhak et al.'s scheme [17,18]. To achieve clear comparisons, let P be the point multiplication in ECC, M be the point multiplication in the multiplicative cyclic group, G be the modular exponentiation in the multiplicative cyclic group, B be the bilinear pairing operation, S be the algebraic signature, E be the symmetric encryption, D be the symmetric decryption, c is the number of challenging blocks in each auditing query, and k be the number of DCTs. Table 2 details the comparisons among these schemes.

In the tag generation phase, scheme [14] spends $(nS + nE)$ for the tags generation and tag encryption. Scheme [15] and scheme [16] cost $(nM + nG)$ and $(2nM + nG)$ to calculate the data tags, respectively. Scheme [17,18] costs $2nS$ to generate data block tags and auxiliary tags. The auxiliary tags are used to resist replay attack, but it does not work from the verification process. For the proposed scheme, the costs of tags generation are nS, which are acceptable for the users. We neglect the computational costs of hash operation and addition operation. In the challenge-verification phase of the scheme, verifying proof is $(1S + 2P)$. The speed of point multiplication in the ECC is relatively fast. So, it is acceptable for the TPA (the illustration given by the experiment later). Furthermore, as shown in Table 2, schemes of [15–18] support modification, insertion and deletion while the main objective of [14] is remote data possession checking based on the algebraic signature. Although Sookhak et al.'s scheme [17,18] outperforms [15,16] in tag generation and proof verification, it consumes more when performs insertion and deletion. To insert a new block or delete an existing block, it must move forward or backward the remaining $(n/k - i)$ blocks of the DCT, which contains (n/k) blocks. Obviously, in terms of computational complexity, the proposed scheme performs better than Sookhak et al.'s scheme [17,18] if we want to update data blocks.

Table 2 details the comparisons.

Table 2. Comparisons of computational complexity.

Schemes	Computational Costs				
	Tag Generation	**Proof Verification**	**Modification**	**Insertion**	**Deletion**
Chen [14]	$nS + nE$	$c(D + S)$	—	—	—
Tian [15]	$nM + nG$	$(c+1)M + 1G + 1B$	$O(c)$	$O(c)$	$O(c)$
Shen [16]	$2nM + nG$	$(c+1)G + 3B$	$O(c)$	$O(c)$	$O(c)$
Sookhak [17,18]	$2nS$	$1S$	$O(c)$	$O(n/k + c)$	$O(n/k + c)$
Our scheme	nS	$1S + 2P$	$O(c)$	$O(c)$	$O(c)$

Note: "—" means not available.

To demonstrate the efficiency, we further conducted simulations in MATLAB 2014 and Eclipse 2012 on a machine with Dual Core 2.60 GHz. As shown in Table 1, schemes [15,16] perform multiple modular multiplication operations to generation tags from Table 2, which has more computation costs than our scheme. Meanwhile, since the number of verification for scheme [14] is finite, we prefer simulating the comparison of our scheme with scheme [17,18]. For consistency, the 160-bit order elliptic curve is applied for both schemes, and the experiment results are the average of the running time.

Figure 7 shows the computational costs of tag generation, where the size of the block is 8KB, and the number is from 100 to 1000. The experimental results of Figure 7 are in agreement with the numerical analysis.

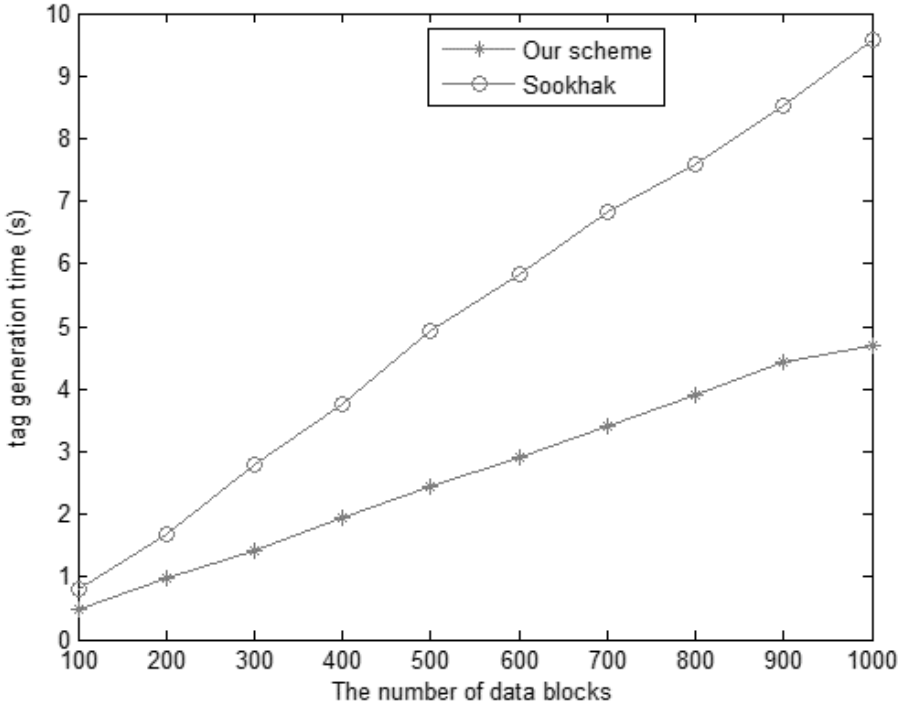

Figure 7. The computational time of generating data block tags.

The computational overheads of proof verification with the number of the challenged blocks from 50 to 500 are shown in Figure 8. We add two-point multiplication operations in the verification process. However, the time difference is less than $1ms$ compared with the scheme [17,18], which is negligible in actual verification. Therefore, the proposed scheme has an acceptable computational cost while achieving higher security, which is very practical for the actual situation.

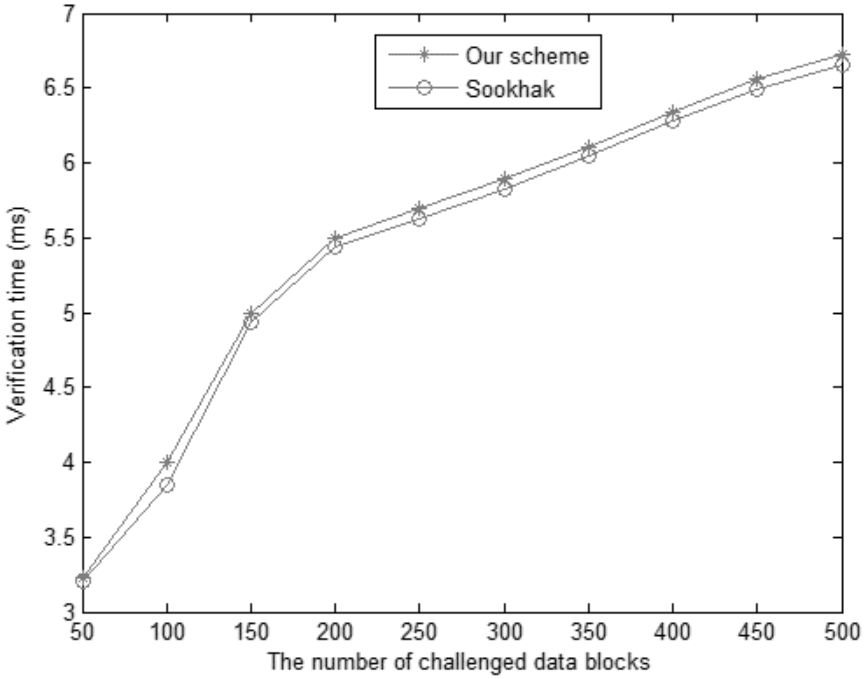

Figure 8. The verification time for different challenged data blocks.

For dynamic updating, the computational costs are smaller than the scheme [17,18] in Figure 9. When the number of updated data blocks is 500, the time of scheme [17,18] is nearly 300 ms, which is almost three times that of the proposed scheme. Hence, we suggest that the proposed data structure is more effective.

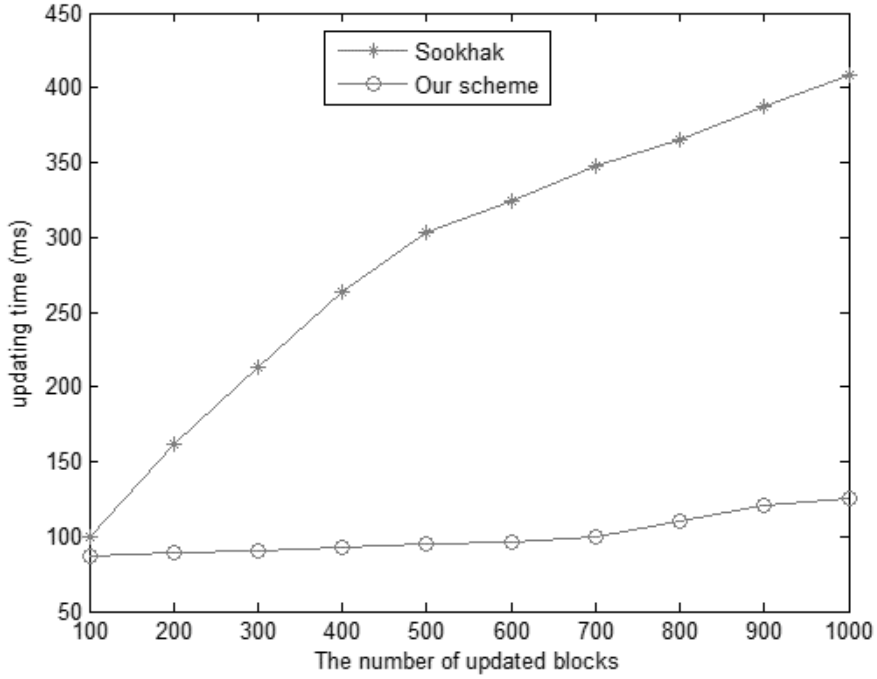

Figure 9. The updating time for different updated data blocks.

8. Conclusions

Towards making several attacks never hurt the cloud storage, in this paper, we propose a public data verification scheme that ensures efficient verification and resists the malicious attacks.

Unlike traditional solutions, the proposed scheme introduces symmetric encryptions to protect data privacy that is critical and suitable for practical use. Meanwhile, a novel data structure named DCHL is designed to support dynamic updating with low communication and computational costs. Theoretical analysis shows that the TPA can detect the corrupted data blocks with a high probability, even though the CSP conducts the misbehavior. Furthermore, real experimental results make the proposed scheme substantially more convincing.

Moreover, no single method can achieve perfect data integrity verification for various scenarios. Due to the query, the complexities of the operations on a linked list are not constant in different scenarios. Frequent data changes on a massive number of data blocks put forward higher requirements for the query performance. Therefore, how to improve the efficiency with specially designed data structures is expected to be tackled in the future.

Author Contributions: Conceptualization, Y.P. and B.W.; methodology, Y.P., Y.Z., K.L. and B.W.; validation, B.W.; formal analysis, Y.P., Y.Z., and B.W.; investigation, B.W.; resources, Y.Z. and K.L.; writing—original draft preparation, Y.Z. and K.L.; writing—review and editing, Y.P. and B.W.; supervision, B.W.; project administration, Y.P. and B.W.; and funding acquisition, Y.P. and B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key R&D Program of China under Grant No. 2017YFB0802000, the National Natural Science Foundation of China under Grant Nos. U19B2021, U1736111, the National Cryptography Development Fund under Grant No. MMJJ20180111, the Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04, the Program for Science and Technology Innovation Talents in Universities of Henan Province under Grant No. 18HASTIT022, the Fundamental Research Funds for the Central Universities, and the Innovation Fund of Xidian University under Grant No. 5001-20109195456, the Innovation Scientists and Technicians Troop Construction Projects of Henan Province.

Acknowledgments: The authors would like to thank the Associate Editor and the anonymous reviewers for their constructive comments that greatly improved the quality of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript or in the decision to publish the results.

References

1. Grobauer, B.; Walloschek, T.; Stocker, E. Understanding Cloud Computing Vulnerabilities. *IEEE Secur. Priv.* **2011**, *9*, 50–57. [CrossRef]
2. Wu, T.; Yang, G.; Mu, Y.; Chen, R. Privacy-enhanced Remote Data Integrity Checking with Updatable Timestamp. *Inf. Sci.* **2020**, *527*, 210–226. [CrossRef]
3. 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]
4. 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, Alexandria, VA, USA, 29 October–2 November 2007; pp. 598–609.
5. Erway, C.C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic Provable Data Possession. *ACM Trans. Inf. Syst. Secur. TISSEC* **2015**, *17*, 15. [CrossRef]
6. Wang, C.; Chow, S.S.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving Public Auditing for Secure Cloud Storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375. [CrossRef]
7. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving Public Auditing for Data Storage Security in Cloud Computing. In Proceedings of the 29th IEEE Conference on Computer Communications (Inforcom), San Diego, CA, USA, 14–19 March 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1–9.
8. Yang, K.; Jia, X. An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1717–1726. [CrossRef]
9. Yang, K.; Jia, X. Data Storage Auditing Service in Cloud Computing: Challenges, Methods and Opportunities. *World Wide Web* **2012**, *15*, 409–428. [CrossRef]
10. Zhu, Y.; Ahn, G.J.; Hu, H.; Yau, S.S.; An, H.G.; Hu, C.J. Dynamic Audit Services for Outsourced Storages in Clouds. *IEEE Trans. Serv. Comput.* **2013**, *6*, 227–238.

11. Zhu, Y.; Hu, H.; Ahn, G.J.; Yu, M. Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2231–2244. [CrossRef]
12. Shacham, H.; Waters, B. *Compact Proofs of Retrievability. Asiacrypt*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5350, pp. 90–107.
13. Boneh, D.; Lynn, B.; Shacham, H. Short Signatures From The Weil Pairing. *J. Cryptol.* **2004**, *17*, 297–319. [CrossRef]
14. Chen, L. Using Algebraic Signatures to Check Data Possession in Cloud Storage. *Future Gener. Comput. Syst.* **2013**, *29*, 1709–1715. [CrossRef]
15. Tian, H.; Chen, Y.; Chang, C.C.; Jiang, H.; Huang, Y.; Chen, Y.; Liu, J. Dynamic-hash-table based Public Auditing for Secure Cloud Storage. *IEEE Trans. Serv. Comput.* **2017**, *10*, 701–714. [CrossRef]
16. Shen, J.; Shen, J.; Chen, X.; Huang, X.; Susilo, W. An Efficient Public Auditing Protocol with Novel Dynamic Structure for Cloud Data. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2402–2415. [CrossRef]
17. Sookhak, M.; Gani, A.; Khan, M.K.; Buyya, R. Dynamic Remote Data Auditing for Securing Big Data Storage in Cloud Computing. *Inf. Sci.* **2017**, *380*, 101–116. [CrossRef]
18. Sookhak, M.; Akhunzada, A.; Gani, A.; Khan, M.K.; Anuar, N.B. Towards Dynamic Remote Data Auditing in Computational Clouds. *Sci. World J.* **2014**, *380*, 269357. [CrossRef] [PubMed]
19. Juels, A.; Kaliski, B.S., Jr. PORs: Proofs of Retrievability for Large Files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 584–597.
20. 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 Netowrks, New York, NY, USA, 22–25 September 2008; p. 9.
21. Krzywiecki, L.; Kutylowski, M. *Proof of Possession for Cloud Storage via Lagrangian Interpolation Techniques*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 305–319.
22. Sebé, F.; Domingo-Ferrer, J.; Martinez-Balleste, A.; Deswarte, Y.; Quisquater, J.J. Efficient Remote Data Possession Checking in Critical Information Infrastructures. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 1034–1038. [CrossRef]
23. Worku, S.G.; Xu, C.; Zhao, J.; He, X. Secure and Efficient Privacy-preserving Public Auditing Scheme for Cloud Storage. *Comput. Electr. Eng.* **2014**, *40*, 1703–1713. [CrossRef]
24. Yu, Y.; Au, M.H.; Ateniese, G.; Huang, X.; Susilo, W.; Dai, Y.; Min, G. Identity-based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 767–778. [CrossRef]
25. Ge, X.; Yu, J.; Zhang, H.; Hu, C.; Li, Z.; Qin, Z.; Hao, R. Towards Achieving Keyword Search over Dynamic Encrypted Cloud Data with Symmetric-Key Based Verification. *IEEE Trans. Dependable Secur. Comput. Early Access* **2020**, 1–16. [CrossRef]
26. Shen, J.; Tan, H.W.; Wang, J.; Wang, J.W.; Lee, S.Y. A Novel Routing Protocol Providing Good Transmission Reliability in Underwater Sensor Networks. *J. Internet Technol.* **2015**, *16*, 171–178.
27. Chen, X.; Li, J.; Weng, J.; Ma, J.; Lou, W. Verifiable Computation over Large Database with Incremental Updates. *IEEE Trans. Comput.* **2016**, *65*, 3184–3195. [CrossRef]
28. Sun, Y.; Liu, Q.; Chen, X.; Du, X. An Adaptive Authenticated Data Structure With Privacy-Preserving for Big Data Stream in Cloud. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3295–3310. [CrossRef]
29. Chen, X.; Li, J.; Huang, X.; Ma, J.; Lou, W. New Publicly Verifiable Databases with Efficient Updates. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 546–556. [CrossRef]
30. Odlyzko, A.M. Discrete Logarithms in Finite Fields and Their Cryptographic Significance. In *Workshop on the Theory and Application of of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1984; pp. 224–314.
31. Litwin, W.; Schwarz, T. Algebraic Signatures for Scalable Distributed Data Structures. In Proceedings of the 20th International Conference on Data Engineering, Boston, MA, USA, 2 April 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 412–423.