Dynamic remote data auditing for securing big data storage in cloud computing

Mehdi Sookhak\textsuperscript{a,}\textsuperscript{*}, Abdullah Gani\textsuperscript{a}, Muhammad Khurram Khan\textsuperscript{b}, Rajkumar Buyya\textsuperscript{c}

\textsuperscript{a} Center for Mobile Cloud Computing Research (C4MCC), University of Malaya, Kuala Lumpur, Malaysia
\textsuperscript{b} Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 12372, Saudi Arabia
\textsuperscript{c} Department of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia

\section{Introduction}

Nowadays, organizations produce a huge amount of sensitive data, such as personal information, financial data, and electronic health records. Consequently, the amount of digital data produced has increased correspondingly and often overwhelmed the data storage capacity of many organizations. The management of such a large amount of data in local storage system is difficult and incurs high expenses because of high-capacity storage systems needed and the expert personnel to manage them. Although the cost of storage hardware has tremendously decreased in recent years, about 75\% of the total ownership cost is still attributed to management data storage [3,13,21,41,49]. The emerging cloud computing paradigm provides a convenient, pay-as-you-go, on-demand network access to a shared pool of configurable computing resources, and demands minimum service provider...
interaction or management effort. Organizations now have an option to outsource their data to cloud storage to decrease the burden on local data storage and also to reduce maintenance cost [4,29,34,42,56].

Although Clouds offer tangible benefits to data owners, outsourcing data to a remote server and delegating management of data to an untrusted cloud service provider, can lead to loss of physical control over the data [17,30,33,57,60]. To the clients, the cloud is inherently neither secure nor reliable and this poses new challenges to the confidentiality, integrity, and availability of data in cloud computing. For example, deleting less frequently retrieved data to provide free disk space or concealing damaged or compromising data to protect the reputation of the organization [3,47,52,61,67]. Some organizations have reported data corruption in servers of major cloud infrastructure providers, and there had been many instances of cloud service outages, such as, Amazon S3 breakdown, Gmail email mass deletion, Sidekick Cloud Disaster, and Amazon EC2 service’s outage [5,50]. The Privacy Rights Clearinghouse (PRC) [1] reported that more than 535 data breaches happened in 2011—breach of cloud-based email service providers in Epsilon, compromise of Sony PlayStation Network, Sony Online Entertainment and Sony Pictures, theft of customers’ information on EMC’s RSA, and theft of 3.3 million patients’ medical data of Sutter Physicians Services. When adversaries gain control over the cloud server, they have the capability to launch forge attack or replay attack which is aimed at breaking the linear independence among encoded data by replacing the data stored in the corrupted cloud server with old encoded data. Therefore, the integrity of users’ data stored on the remote cloud server is vulnerable to internal and external attacks.

Without a local copy of the data, traditional integrity verification techniques such as hash functions and signatures are inapplicable in the cloud storage. Also, it is impossible to download a large-size file from the cloud storage [2,8,40]. The situation is made worse when users access data using their mobile devices. In this context, a more efficient technique is required to remotely verify the integrity of the outsourced data in the cloud. To address the issue of data integrity in cloud computing, researchers have developed the Remote Data Auditing (RDA) techniques, which can securely, frequently, and efficiently validate the proof of data possession by generating a random challenge [6,16], RDA techniques are divided into three main categories: (1) Integrity-based: the auditor in this group is only permitted to check on the integrity of the outsourced data directly or through a third party; (2) Recovery-based: aside from verifying data integrity, the techniques in this category support Forward Error Correcting (FEC) codes by leveraging on the Reed-Solomon erasure-correcting code [38], and (3) Deduplication-based: it ensures data integrity and efficiency by removing data redundancy and increasing data storage optimization [49]. To design and implement a reliable remote data auditing method, the following pertinent properties must be taken into consideration:

1. **Efficiency:** audit data with minimum computational cost, storage cost, and communication cost between client and server;
2. **Public/private verifiability:** in the private verification mode, the data owner is only able to check the integrity of the outsourced data, while in the public verification mode, the intricate task of verification is delegated to a third party to reduce the computational cost of data auditing for the data owner;
3. **Frequency:** repeating the verification process as frequently as possible with different challenge messages;
4. **Probability of detection:** it represents the probability of detecting potential data corruption; and
5. **Dynamic update:** ability to perform insert, delete, modify, and append operations on the outsourced data without having to download all the data [48,49,52,62].

The existing RDA methods require frequent auditing and involve many processes and frequent data transmission. Consequently, the RDA methods incur additional computational and communication costs on the auditor, which is a big burden for many data owners, especially when they use mobile devices that have limited computing resources (CPU). On the other hand, the main design principle of the RDA methods is to support dynamic data update operations for different applications due to the dynamic nature of the data. Therefore, different types of data structure such as binary tree are used in such methods to achieve this goal. However, the applied data structures in the RDA methods are unable to effectively support dynamic data update operation for large-scale data efficiently, especially frequent data update. This is because the auditor must rebalance a large number of data blocks within the data structure many times, and this incurs high computational cost on the auditor. In this context, it is imperative to design a new data structure to support dynamic update for large-scale data.

We propose an efficient remote data auditing method for securing big data storage in cloud computing based on an algebraic signature. This signature allows the auditor to check data possession in cloud storage, and this incurs fewer computational overheads on the auditor and server in comparison to homomorphic cryptosystem. Furthermore, we can extend our data auditing method by designing a data structure that allows the auditor to perform dynamic data update operation efficiently with minimum computational overhead on the client and cloud server. The contributions of this paper can be summarized as follows:

1. Development of efficient remote data auditing scheme for data storage in cloud computing based on the algebraic signature. The scheme incurs minimum computational and communication costs on the auditor and server side.
2. Design of a new data structure—Divide and Conquer Table (DCT)—to efficiently support dynamic data operations such as insert, append, delete, and modify. With the new data structure, our method can be applied for frequent update of large-scale data with minimum computational cost on the auditor and server.
3. Implemented proposed remote data auditing method in a real environment and demonstrated its ability to provide better data security and performance when compared to state-of-the-art data auditing methods.

The rest of the paper is organized as follows. Section 2 discusses the related works in this area, and Section 3 introduces the fundamental concepts applied in our proposed method. In Section 4, we introduce the details of our remote data auditing scheme. We describe the security analysis of our scheme in Section 5, and Section 6 presents an optimization algorithm of DCT.
divisions. Section 7 discusses the performance analysis in terms of computational overhead. Finally, the conclusion of this paper is presented in Section 8.

2. Related work

Recently, there has been a great deal of attention on the use of the RDA schemes to check the correctness of outsourced data in cloud computing [7,8,19,51,53–55,61]. We review the existing RDA schemes used in verifying data integrity in single cloud server, and discuss the advantages and disadvantages of such methods.

Ateniese et al. [7] were the first to propose the provably-secure scheme to verify the integrity of data storage in the cloud, without having to download all the data. The authors used the RSA-based homomorphic Verifiable Tag (HVT) to generate a single tag by combining the block tags. HVT also allows a server to construct a proof and permits the client to check whether the server has certain blocks, even though the client may not have access to the blocks. However, this method incurs high server computational or communication costs for the whole file due to the use of RSA numbering [31,45].

Juels and Burton S. Kaliski [28] implemented a new type of RDA techniques—Proofs of Retrievability (POR)—to validate data integrity and to prevent data corruption by using forward error-correcting codes, remotely. The POR method relies on a set of random-value check blocks known as sentinels which are generated by using a one-way function. The auditor then randomly inserts the sentinel blocks into the blocks before uploading them to the server. However, the number of queries in this method is dependent on the number of sentinel blocks. The POR method incurred high computational overhead on the client side, which resulted from having to perform the error recovery and data encryption processes. Shacham and Waters [46] introduced a method to provide efficiency and security for the POR method based on the BLS homomorphic authentication technique [9]. The BLS technique allows the auditor to aggregate the tags into a fixed size in order to minimize the network computational overhead, and uses the Reed-Solomon code to recover the errors. However, this method is not applicable in interactive proof systems because an adversary can gain access to the outsourced data by using the public verification protocol.

Supporting dynamic data update efficiently is an important issue in the majority of the remote data auditing methods in which the auditor is able to dynamically update the outsourced data without having to retrieve the outsourced file. In [8], a new RDA method was proposed to improve the scalability and efficiency of the data possession scheme. The authors used the symmetric-key operations to overcome the problem of static RDA methods for updating data dynamically. In this method, the data owner pre-computes a number of short-ownership verification tokens before uploading the data to the cloud. Although this method supports the modify, delete, and append operations, updating a data block can result in the re-computation of all the remaining tokens, and this will incur high computational cost on the data owner, and is therefore impractical for large files.

Erway et al. [19,20] leveraged a cross between an original skip list [39], an authentication dictionary [23] and a rank-based information to design a fully dynamic auditing method. This method can check the integrity of data blocks of variable-sizes but it is unable to verify the integrity of individual block [66].

Most POR methods cannot efficiently support dynamic data update because the server is unable to establish a relationship between the data blocks and the encrypted code-words. Cash et al. [14] addressed this issue by designing the first dynamic POR scheme based on the ORAM technique [22]. The dynamic POR method also incurs high computational overhead on the client and server side.

Wang et al. [55] proposed a dynamic remote data auditing method by combining the Merkle Hash Tree (MHT) [36] with the bilinear aggregate signature [9]. The main contribution of this method is the modification of the MHT structure by sorting the leaf nodes from left to right which will help in identifying the position of the updated block by following this sequence to the root and computing the root of the modified MHT. However, this method is vulnerable to leakage of data to the auditor and will incur high computational cost on the auditor, especially for large-scale files.

To overcome the privacy issue in [55], Yang and Jia [61] designed an efficient data auditing scheme using the bi-linearity property of the bilinear pairing to generate an encrypted proof that can only be verified by the auditor. They also proposed a data structure to support dynamic data update operations in which the data owner needs to store block index and block logical location for each block of the outsourced file. However, to delete or insert a data block (i), the auditor has to find the position of the block and shift the remaining blocks (n – i) to create or delete a row in the data structure. Moreover, increasing the number of outsourced blocks can cause the auditor to shift a huge number of blocks, and this will incur high computational cost. The other disadvantage of this method is that it is unable to efficiently support frequent dynamic update operations efficiently due to the node re-balancing problem, and this will also incur high computational overhead on the auditor side. Furthermore, bilinear pairing computation is more expensive than the algebraic structure that is used in our method [15].

Wei et al. [57,58] considered the issue of outsourced computation security in cloud computing to ensure whether the CSP performs the necessary computations. They proposed a privacy preserving and computational auditing method, SecCloud, that uses Commitment-Based Sampling (CBS) technique [18] and designated verifier signature [24,65] to achieve privacy cheating discouragement and thus, minimize the computational cost in cloud computing. To reduce the communication and computational costs, a batch verification algorithm was suggested for handling various users’ requests, concomitantly. They also developed a secure-aware cloud computing environment to implement SecCloud in the real world environment [57]. Li et al. [32] extended this method to support both data integrity and deduplication by using the convertible encryption technique.

With the advent of cloud computing, most software applications have shifted to cloud storage instead of local storage of data. However, cloud storage is vulnerable to various security breaches, privacy abuses, and access control violations, which
threaten the copyright of these applications. Yu et al. [63] addressed the copyright-infringement issue in the cloud by developing a watermarking method for protecting the outsourced software applications.

The existing data auditing methods assume that the data owner’s secret key is secure, but this is not always true. Yu et al. [26] proposed an auditing method to mitigate the damage of the data owner’s key exposure issue, in which the secret key of the data owner is updated by employing the binary tree structure and the pre-order traversal technique.

Some methods [11,27,64] have addressed the issue by designing an efficient and secure user revocation authentication scheme. In [64], new polynomial-based and proxy-based authentication tags were used to aggregate the authentication tag and delegate user revocation operations. In [11], the proxy re-signature is used by the CSP to sign the outsourced data block instead of downloading the whole file and having it re-signed by the data owner. Jian et al. [27] considered the problem of collusion attack and secure revoked group users during user revocation in the existing method. The main idea behind this method is the use of Asymmetric Group Key Agreement (AGKA) [59] and group signatures [10] to enable each user of the group to encrypt/decrypt the data blocks and prevent collusion attack.

3. Preliminaries

This section presents the background of our dynamic remote data auditing method. The general architecture of the remote data auditing protocol is explained, followed by description of the algebraic signature technique that is used to implement the proposed method.

3.1. System model

There are four main entities in the architecture of the proposed RDA method: (1) Data Owner (DO): the person, enterprise, or businesses which upload their data to the cloud space, and might later update the outsourced data by performing the modify, delete, insert, and append operations; (2) Cloud Storage Provider (CSP): has a considerable amount of computing resources and storage systems and is responsible for managing the cloud servers and Do’s data, (3) Third Party Auditor (TPA): helps to reduce the computational burden of the auditing process on the DO, as it has adequate skills and competency to perform the auditing task; and (4) User: has to be authenticated by the DO as a trusted user and be permitted to have predetermined type of access to the outsourced data. Fig. 1 shows the main components of RDA methods and their interactions.

3.2. Algebraic signatures

The algebraic signature is a type of hash function with algebraic properties that computes the signatures of unseen messages in a limited way. The main property of the algebraic signature technique is to take a signature of the sum of some random blocks, and produce the same result as when taking the sum of the signatures of the corresponding blocks [44]. Therefore, the algebraic signature of file $F$ consists of $n$ data blocks ($f[1], f[2], \ldots, f[n]$) is computed by:

$$S_{\gamma}(F) = \sum_{i=1}^{n} f[i].\gamma^{i-1}$$

(1)

where $\gamma$ is an element in the Galois field and composed of a vector of various non-zero elements $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$. A number of algebraic signature properties are explained as follows.

**Proposition 1.** The algebraic signature of concatenation of block $f[i]$ with length $r$ and $f[j]$ is computed by [35]:

$$S_{\gamma}(f[i] || f[j]) = S_{\gamma}(f[i]) \oplus r^j S_{\gamma}(f[j])$$

(2)
Proposition 2. The signature of summation of several blocks of a file \( F \) is equal to summation of the signature of each of the blocks.

\[
S_r(f[1]) + S_r(f[2]) + \cdots + S_r(f[m]) = S_r(f[1] + f[2] + \cdots + f[m])
\]  

(3)

Proof. Assume that File \( F \) is divided into \( m \) blocks and each block consists of \( n \) sectors. Then:

\[
S_r(f[1]) + S_r(f[2]) + \cdots + S_r(f[m]) = \sum_{j=1}^{n} f[1][j] \gamma^{j-1} + \sum_{j=1}^{n} f[2][j] \gamma^{j-1} + \cdots + \sum_{j=1}^{n} f[m][j] \gamma^{j-1}
\]

\[
= \sum_{j=1}^{n} \gamma^{j-1} (f[1][j] + f[2][j] + \cdots + f[m][j])
\]

\[
= S_r(f[1] + f[2] + \cdots + f[m])
\]

where \( f[i][j] \) indicates the \( j \)th bit of block \( i \) in file \( F \).

Proposition 3. The algebraic signature of summation of two files, \( F \) and \( G \), is equal to the summation of signature of such files.

\[
S_r(F + G) = S_r(F) + S_r(G)
\]

(4)

Proof. The summation of signature of the two files, \( F \) and \( G \) consist of \( n \) blocks, is equal to:

\[
S_r(F) + S_r(G) = \sum_{i=1}^{n} f[i] \gamma^{i-1} + \sum_{i=1}^{n} g[i] \gamma^{i-1}
\]

\[
= \sum_{i=1}^{n} \gamma^{i-1} (f[i] + g[i])
\]

\[
= S_r(F + G)
\]

4. The proposed scheme

We present techniques and algorithms of our dynamic remote data auditing scheme. We also present the proof of correctness of our method by using the characteristics of the algebraic signature technique.

4.1. Remote data auditing algorithm

Assume that the input file \( F \) is divided into \( m \) data blocks by using the data fragment technique where each of the blocks involves \( n \) sectors. Based on the data fragment technique, if the number of sectors of the last block is less than \( n \), we have to augment the size of the block by setting \( f[m][j] = 0 \) for \( j \leq n \). The proposed remote data auditing scheme consists of the following steps:

Setup: The DO must first generate the public key and secret key by using the KeyGen algorithm (KeyGen(1\(^k\) → (pk, sk)), where \( k \) is a security parameter. Following this, the unique tag (metadata) for each block of input file is computed based on the algebraic signature of the block by using Eq. (5).

\[
T_i = S_r(f[i]|(ID_F||i||L_i||V_i))
\]  

(5)

where \( f[i] \) is the \( i \)th block of file \( F \), ID\(_F\) is the unique identity of the file, \( L_i \) is the logical number of the file in the DCT table, and \( V_i \) indicates the version of the data block. To prevent replay attack, the DO must compute \( C_i = S_r(ID_F||i||L_i||V_i) \) for each data block, where \( i \) is the index of the block. After generating the tags for all blocks, the DO uploads the data blocks along with the corresponding tags to the cloud\([f[i], T_i, C_i], i = 1^n\) and delete the local copy of the file.

Challenge: To verify the integrity of the outsourced data blocks, the DO is required to generate a challenge message. This message consists of \( c \) data blocks randomly as a challenge message \((chal = \{cs_i\}_{i=1}^{c})\) by using pseudo-random permutation \([25]\) keyed with a fresh randomly-chosen key to prevent the server from anticipating the block indices.

Proof. When the challenge message is received by the cloud server, the proof message, including a linear combination of the blocks \( (\sigma) \) and the aggregation of authenticator tags \( (\mu) \), is generated based on the challenge message and the corresponding tags by using Eq. (6).

\[
\mu = \sum_{i = cs_1}^{cs_c} T_i \oplus C_i
\]

\[
\sigma = \sum_{i = cs_1}^{cs_c} f[i]
\]  

(6)

Verification: Upon receiving the pair \((\mu, \sigma)\) as proof, the DO verifies the integrity of the blocks based on the algebraic signature of the block tags using the formula:

\[
S_r(\sigma) \equalplus \mu
\]  

(7)
To improve the security of the proposed method, the DO can sign the file id by using the DO’s private key in the setup step and subsequently verify the signature in the verification step by using the DO's public key.

4.2. Dynamic data operations

Dynamic data update—an essential feature of the data auditing methods—allows data owners to update their outsourced data, whenever necessary without the need to download the data. However, most of the existing methods are unable to support this feature. In this section, we describe our proposed data structure—Divide and Conquer Table (DCT)—to perform dynamic update operations efficiently. The DCT also prevents the server from carrying out the replay attack in which the previous version of the stored data in the server is used to pass the verification phase, rather than the updated version.

The DCT includes two important components: (1) logical index ($L_i$): that indicates the original index of the data block; and (2) version number ($V_i$): that indicates the current version of the block based on the number of updates. If the DO updates a data block, the corresponding $V_i$ in DCT must be incremented by 1 to show the modified block. It is clear that the physical position of the outsourced data block matches with the index of each block in DCT. The DO is responsible for generating the DCT data structure for each file before outsourcing it to the cloud. Also, the DO is able to manage the DCT during an update operation or delegate this task to the TPA.

Although this data structure is able to accomplish dynamic update operation for a small file efficiently, managing dynamic update for large-scale files incurs expensive computational cost on the auditor due to node re-balancing issue. For example, to insert a new data block after the $i$th block of the file, the auditor must shift $n-i$ blocks, and this incurs additional computational overhead on the auditor. To address this issue, we divide the DCT into $k$ separate data structures with size of $\frac{n}{k}$ in order to reduce the size of the DCT and to alleviate node rebalancing. Therefore, to insert a new block after the $i$th block using the new DCT structure, the DO is only required to shift $\frac{n}{k} - i$ blocks. The experimental results show that the proposed data structure can manage dynamic data update operation for large-scale files efficiently. In the rest of this section, we discuss how our scheme performs dynamic data operations, such as modify, insert, delete, and append.

4.2.1. Data modify

Data modification is one of the important requirements of data auditing techniques to allow the DO to update the outsourced file by altering a number of blocks without having to download all the blocks. To modify the $i$th block of the file ($f[i]$) to $f'[i]$ using the proposed method, the DO runs the modification algorithm as follows:

1. Find the location of the requested block ($i$) in the DCTs by comparing $i$ with the range of each of the DCTs and then the version number of the block must be increased $V_i = V_i + 1$.
2. Generate a new tag for the modified data block by using Eq. (8):

$$T_i' = S_f(f'[i] || (ID_F[i] || V_i'))$$
$$C_i' = S_f(ID_F[i] || L_i || V_i')$$

3. Prepare a modification message, including ($ID_F, i, f'[i], T_i', C_i'$), and transfer it to the CSP.

When the modification message is received by the CSP, the block $f[i]$ is replaced with $f'[i]$ and the tag of the data block ($T_i, C_i$) is also changed to ($T_i', C_i'$). Fig. 2 shows the effect of data modification on the DCTs when the data owner modifies block $f[7]$ (the number of entities in each DCT is 5).

4.2.2. Data insert

To insert a new data block ($f'[i + 1]$) after the $i$th block of the file ($f[i]$), the DO must execute the insert algorithm that performs the following tasks:

![Fig. 2. Effects of modifying $f[7]$ on DCTs when the number of blocks in each table is 5.](image)
1. Find the DCT that stores the $i$th block of the file $f$ based on the range of DCTs and identify the precise location of the new block ($l$) in this data structure.

2. Shift the subsequent blocks ($\frac{l}{2} - l$) one position down, and create a new row ($L^*_{i+1}, V^*_{i+1}$) after the $i$th block.

3. Set the original index of the new data block $L^*_{i+1} = m + 1$ and the version number of the new block $V^*_{i+1} = 1$ where $m$ is the maximum number of blocks in the DCTs.

4. Increase the maximum range of the current DCT along with the minimum and maximum ranges of subsequent DCTs by 1.

5. Generate a tag ($T^*_{i+1}, C^*_{i+1}$) for the new data block by using Eq. (9):

$$
T^*_{i+1} = S_f(f^*[i+1][\text{ID}_f][i+1][L^*_{i+1}][V^*_{i+1}]),
$$

$$
C^*_{i+1} = S_f(\text{ID}_f[i+1][L^*_{i+1}][V^*_{i+1}])
$$

6. Prepare the insert message, including ($\text{ID}_f, i + 1, f^*[i+1], T^*_{i+1}, C^*_{i+1}$), and send it to the CSP.

Upon receiving the insert message, the CSP inserts this new block and the corresponding tag after position $i$ in the file. Fig. 3 illustrates the effect of insert operations on the DCT structure. To insert a new block after the 7th block of the file, the DO only needs to shift three entities down to insert this new block ($DCT_2[3] = [16, 1]$) in the second DCT and increase the maximum and minimum ranges of the next DCTs along with the maximum range of DCT2.

### 4.2.3. Data append

The append operation allows the data owner to insert a new data block to the end of the file. Therefore, the DO only needs to insert a new row ($L^*_{n+1}, V^*_{n+1}$) to the end of the last DCT and increase the maximum range of the last DCT without having to shift any entities of the DCTs. Following this, the tag of this block is computed by using Eq. (9), where $i = n$ and the append message including ($\text{ID}_f, n + 1, f^*[n+1], T^*_{n+1}, C^*_{n+1}$) is sent to the CSP. Fig. 4 shows the effect of the append operation on the DCT structure in which the data owner attaches a free row to the end of the third DCT ($DCT_3[6]$), and increases the maximum range of this structure by one (Max Rang$_3 = \text{Max Rang}_3 + 1$).

### 4.2.4. Data delete

The delete operation reverses the insert operation wherein the $i$th block of the outsourced file ($f[i]$) is removed. Therefore, to delete a data block, the DO, firstly, has to find the DCT that stores the $i$th block based on the maximum and minimum ranges

---

Please cite this article as: M. Sookhak et al., Dynamic remote data auditing for securing big data storage in cloud computing, Information Sciences (2015), http://dx.doi.org/10.1016/j.ins.2015.09.004
of the DCTs. The position of such a block \((p)\) is then found in the identified DCT and the block is removed by shifting all the subsequent blocks \( \left( \frac{4}{\text{num blocks}} - p \right) \) one position up. Finally, the DO transfers a delete message including \((\text{ID}_{DCT}, i)\) to the CSP. Fig. 5 shows the impact of the delete operation on DCTs in which the 4th data block \((f[4])\) of the outsourced file is deleted by shifting a single row \((f[5])\) up and reducing the maximum and minimum ranges of DCT2 and DCT3 along with the maximum range of the first DCT.

### Fig. 5. Effects of deleting the \(f[4]\) on DCTs when the number of blocks in each table is 5.

#### 5. Security analysis

We present the security and correctness of the proposed remote data auditing method. Based on the algorithm of the proposed method, when the CSP receives the challenge message \((\{cs_i\}_{i=1}^{c_s})\), a pair \((\mu, \sigma)\) as a proof message is generated and transferred to the auditor. We extend \(\mu\) in Eq. (6) by using the properties of algebraic signature as follows:

\[
\mu = \sum_{i=1}^{c_s} c_i \oplus i
\]

\[
= \sum_{i=1}^{c_s} S_{\gamma}(f[i], V_i) = \sum_{i=1}^{c_s} S_{\gamma}(ID_F, V_i)
\]

\[
= \sum_{i=1}^{c_s} S_{\gamma}(f[i], V_i) \oplus S_{\gamma}(ID_F, V_i)
\]

\[
= \sum_{i=1}^{c_s} S_{\gamma}(f[i])
\]

Upon obtaining the proof message from the CSP, the auditor verifies the proof message to ensure the storage correctness by using Eq. (7). We also extend this equation based on the algebraic signature properties to demonstrate the correctness of the verification algorithm as follows:

\[
S_{\gamma}(\sigma) = S_{\gamma}\left( \sum_{i=1}^{c_s} f[i] \right)
\]

\[
= S_{\gamma}(f[cs_1] + \cdots + f[cs_c])
\]

\[
= S_{\gamma}(f[cs_1]) + \cdots + S_{\gamma}(f[cs_c])
\]

\[
= \sum_{i=1}^{c_s} S_{\gamma}(f[i])
\]

\[
= \mu
\]

The proposed scheme relies on the algebraic signature that generates a small entity as a signature for each block, and shows any modifications in the outsourced file. The algebraic signature can also verify a large amount of stored data in the distributed storage systems, and will incur only minimum computational and communication costs [44]. On the other hand, the probability of collision in the algebraic signature is negligible [35]. For example, if the length of the signature is 64 bits, the probability of collision is very small \((2^{-64})\). As a result, the algebraic signature technique is useful for verifying the correctness of outsourced data, especially when the DO uses mobile devices.

**Please cite this article as:** M. Sookhak et al., Dynamic remote data auditing for securing big data storage in cloud computing, Information Sciences (2015), [http://dx.doi.org/10.1016/j.ins.2015.09.004](http://dx.doi.org/10.1016/j.ins.2015.09.004)**
Table 1
Minimum, maximum, and optimized number of DCT tables for the variable-length of the file.

<table>
<thead>
<tr>
<th>File size (GB)</th>
<th>Number of blocks (n)</th>
<th>Number of divisions (k)</th>
<th>Min</th>
<th>Max</th>
<th>Opt.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.000008</td>
<td>124,999</td>
<td>353.5533906</td>
</tr>
<tr>
<td>2</td>
<td>250,000</td>
<td></td>
<td>1.000004</td>
<td>249,999</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>375,000</td>
<td></td>
<td>1.000003</td>
<td>374,999</td>
<td>612.3724357</td>
</tr>
<tr>
<td>4</td>
<td>500,000</td>
<td></td>
<td>1.000002</td>
<td>499,999</td>
<td>707.1067812</td>
</tr>
<tr>
<td>5</td>
<td>625,000</td>
<td></td>
<td>1.000002</td>
<td>624,999</td>
<td>790.569415</td>
</tr>
<tr>
<td>6</td>
<td>750,000</td>
<td></td>
<td>1.000001</td>
<td>749,999</td>
<td>866.0254038</td>
</tr>
<tr>
<td>7</td>
<td>875,000</td>
<td></td>
<td>1.000001</td>
<td>874,999</td>
<td>935.4143467</td>
</tr>
<tr>
<td>8</td>
<td>1,000,000</td>
<td></td>
<td>1.000001</td>
<td>999,999</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1,125,000</td>
<td></td>
<td>1.000001</td>
<td>1124,999</td>
<td>1060.660172</td>
</tr>
<tr>
<td>10</td>
<td>1,250,000</td>
<td></td>
<td>1.000001</td>
<td>1249,999</td>
<td>1118.033989</td>
</tr>
<tr>
<td>20</td>
<td>2,500,000</td>
<td></td>
<td>1</td>
<td>2499,999</td>
<td>1581.491673</td>
</tr>
<tr>
<td>30</td>
<td>3,750,000</td>
<td></td>
<td>1</td>
<td>3749,999</td>
<td>1936.491673</td>
</tr>
<tr>
<td>40</td>
<td>5,000,000</td>
<td></td>
<td>1</td>
<td>4999,999</td>
<td>2236.067977</td>
</tr>
<tr>
<td>50</td>
<td>6250,000</td>
<td></td>
<td>1</td>
<td>6249,999</td>
<td>2500</td>
</tr>
<tr>
<td>60</td>
<td>7500,000</td>
<td></td>
<td>1</td>
<td>7499,999</td>
<td>2738.612788</td>
</tr>
<tr>
<td>70</td>
<td>8750,000</td>
<td></td>
<td>1</td>
<td>8749,999</td>
<td>2958.039892</td>
</tr>
<tr>
<td>80</td>
<td>10,000,000</td>
<td></td>
<td>1</td>
<td>9999,999</td>
<td>3162.27766</td>
</tr>
<tr>
<td>90</td>
<td>11,250,000</td>
<td></td>
<td>1</td>
<td>11,249,999</td>
<td>3354.101966</td>
</tr>
<tr>
<td>100</td>
<td>12,500,000</td>
<td></td>
<td>1</td>
<td>12,499,999</td>
<td>3535.533906</td>
</tr>
</tbody>
</table>

6. Optimizing the number of DCT structures

As mentioned in Section 1, the main contribution of the proposed method is the reduction in computational overhead during the dynamic data update operations. In Yang and Jia [61], to insert or delete a block, the auditor must shift the remaining blocks, but this leads to considerable overhead on the auditor $O(n)$. We proposed a new data structure to overcome this problem by storing the data blocks in $k$ arrays instead of an integrated data structure. As a result, our method only incurs $O(n/k)$ computational overhead on the auditor. In the worst case scenario, our method incurs $O(k)$ to find the location of the block. Therefore, the proposed method is efficient if and only if:

$$k + \frac{n}{k} \leq n \Rightarrow k + \frac{n}{k} - n \leq 0 \Rightarrow k^2 + n - nk \leq 0$$

Since $k \geq 1$, then:

$$k^2 + n - nk \leq 0 \Rightarrow \begin{cases} k_{\text{min}} = \frac{n - \sqrt{n^2 - 4n}}{2} \\ k_{\text{max}} = \frac{n + \sqrt{n^2 - 4n}}{2} \end{cases}$$

Therefore, the optimal number of divisions is computed by using the following formula:

$$1 - \frac{n}{k^2} = 0 \Rightarrow k^2 = n \Rightarrow k_{\text{Opt}} = \sqrt{n}$$

Table 1 shows the minimum, maximum, and optimized number of DCT tables in the proposed method when the size of the outsourced file is between 1 GB and 100 GB, and the size of each block is 4 KB.

7. Performance analysis

We present the methodology adopted in evaluating the performance of the proposed method in the cloud computing environment. We also analyze the probability of misbehavior detection of this scheme. We discuss the computational complexity during the insert, delete, append, and modify operations and the results of the comparison between our methods and two state-of-the-art remote data auditing methods proposed by Yang and Jia [61] and Wang, Wang, Ren, Lou and Li [55].

7.1. Probability of misbehavior detection

The proposed method is implemented based on a random sampling strategy to lessen the workload on the CSP. The sampling technique divides the input file ($F$) into numerous blocks ($m$) and randomly selects a number of blocks ($c$) as a challenge to perform batch processing. Following this, we analyzed the probability of misbehavior detection of the proposed data auditing method based on such technique.
Assuming that the CSP modifies \( y \) blocks out of the \( m \) outsourced blocks, then the probability of corrupted blocks is equal to \( p_y = \frac{y}{m} \). Let \( c \) be the number of blocks that the DO requested to verify the outsourced data in the challenge step, and \( n \) is the number of sectors in each block. Let \( x \) be a discrete random variable that indicates the number of blocks chosen by the DO that matches the modified blocks by the CSP. We compute the probability that at least one of the blocks picked by the DO matches one of the blocks modified by the server, namely \( P_c(x \geq 1) \) as follows:

\[
P_c(x \geq 1) = 1 - P_c(x = 0) \]

\[
= 1 - \left( \frac{m-y}{m} \right) \left( \frac{m-y-1}{m-1} \right) \ldots \left( \frac{m-y-c+1}{m-c+1} \right) 
= 1 - \left( 1 - \frac{y}{m} \right) \left( 1 - \frac{y}{m-1} \right) \ldots \left( 1 - \frac{y}{m-c+1} \right) 
= 1 - \prod_{i=0}^{c-1} \left( 1 - \frac{y}{m-i} \right)
\]  

(10)

On the one hand,

\[
\left( 1 - \frac{y}{m-i} \right) \leq \left( 1 - \frac{y}{m} \right) \Rightarrow \prod_{i=0}^{c-1} \left( 1 - \frac{y}{m-i} \right) \leq \left( 1 - \frac{y}{m} \right)^c \Rightarrow 1 - \prod_{i=0}^{c-1} \left( 1 - \frac{y}{m-i} \right) \geq 1 - \left( 1 - \frac{y}{m} \right)^c \]

(11)

Therefore:

\[(10) \Rightarrow (11) P_c(x \geq 1) \geq 1 - \left( 1 - \frac{y}{m} \right)^c = 1 - (1 - p_y)^c\]

Since each of the blocks consists of \( n \) sectors, the probability, based on sector corruption \( p_s \) is computed by:

\[p_y \geq 1 - (1 - p_y)^n \Rightarrow (1 - p_y)^c \leq ((1 - p_y)^n)^c \Rightarrow 1 - (1 - p_y)^c \geq 1 - (1 - p_s)^nc\]

\[\Rightarrow P_c(x \geq 1) \geq 1 - (1 - p_s)^nc\]

On the other hand:

\[
\left( 1 - \frac{y}{m-i} \right) \geq \left( 1 - \frac{y}{m-c+1} \right) \Rightarrow \prod_{i=0}^{c-1} \left( 1 - \frac{y}{m-i} \right) \geq \left( 1 - \frac{y}{m-c+1} \right)^c 
\Rightarrow 1 - \prod_{i=0}^{c-1} \left( 1 - \frac{y}{m-i} \right) \leq 1 - \left( 1 - \frac{y}{m-c+1} \right)^c
\]

Therefore:

\[P_c(x \geq 1) \leq 1 - \left( 1 - \frac{y}{m-c+1} \right)^c\]

Then, we can conclude that the probability of misbehavior detection is as follows:

\[1 - \left( 1 - \frac{y}{m} \right)^c \leq P_c(x \geq 1) \leq 1 - \left( 1 - \frac{y}{m-c+1} \right)^c\]  

(12)

Suppose the DO divides a 1 GB file into 125,000 blocks with size of 8 KB per block and uploads the blocks to the cloud storage. Fig. 6 shows the number of challenge blocks (\( c \)) that are required to detect the different number of corrupted blocks (\( y \)) when the probability of misbehavior detection is collected from a set of \( P_y = \{0.7, 0.8, 0.9, 0.99, 0.99999\} \). Thus, if the server modifies \( P_y = 0.1 \) of the outsourced blocks (\( m \)), the DO needs to randomly select 98 blocks as a challenge to achieve \( P_y \) of at least 0.99999. Hence, by increasing the number of corrupted blocks, a smaller number of challenge blocks is required to achieve this probability of detection (i.e., only 22 blocks for \( P_y = 0.4 \)).

Fig. 7 shows the number of challenge blocks when the probability of misbehavior detection is between 0.5 and 1, with variable rate of data corruption. Thus, if the server modifies 0.01% of the outsourced blocks, the DO has to randomly select 520 data blocks as a challenge to detect the corrupted blocks with a probability of 0.9899. Also, when the rate of corrupted blocks is more than 0.1%, the minimum number of challenge blocks is used to audit the outsourced data (between 34 and 51 data blocks).

7.2. Evaluation and experimental results

Table 2 shows a comparison between the proposed method and state-of-the-art remote data auditing protocols based on three important parameters: communication cost, computation cost of data auditing, and computation cost of dynamic data update: (1) Communication cost: the attribute of communication cost shows the amount of data transfer between the auditor and the server in different phases of the auditing scheme; (2) Computation cost of data auditing: each step of the data auditing method is responsible for performing a specific task, and this puts some computational burden on the auditor or server. From the verifier’s point of view, computational cost of data auditing indicates the computational resources that are used by the auditor.
to verify the integrity of the outsourced data. From the server’s point of view, the computational cost of data auditing indicates the required time to process and generate the proof message in the response step; and (3) Computation cost of dynamic data update: dynamic data auditing methods allow data owners to update the outsourced data by using insert, delete, append, and modify operations. During dynamic data update operation, the data owner needs to accomplish some tasks such as finding the location of a requested block, generating new tag, and re-balancing the applied data structure based on the update operation. As a result, the required time to execute the update operations—insert, delete, append, and modify—is called the computation cost of dynamic data update.

In the Wang method [55], the maximum computational overhead is incurred during dynamic data update because the MHT data structure is used to check the integrity or perform the update operations on the outsourced data blocks. Although the Yang scheme [61] is efficient ($O(c)$) in modifying and appending a block, to insert a block after $i$ or delete a specific block ($f[i]$), the verifier must shift $(m - i)$ entities in the data structure. As a result, the computational overhead of this method during the insert...
and delete operations is $O(m)$. To address this problem and improve the auditing scheme, we designed a new data structure (DCT) to reduce computational overhead. As mentioned in Section 4.2, to insert or delete a data block, the verifier has to shift a part of the outsourced data blocks ($\frac{c}{m} - 1$) that incurs $O(\frac{c}{m})$ computational overhead on the verifier. It is important to mention that to find a block ($f(i)$) in the DCT structure, the verifier only needs to divide the location of a block to $k$ and find the appropriate DCT that incurs negligible computational overhead on the verifier.

The experiment of the proposed remote data auditing scheme was carried out by using a Eucalyptus private Infrastructure as a Service (IaaS) cloud. Eucalyptus, an acronym for “Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems”, was designed initially at the University of California, Santa Barbara, to support high performance computing (HPC) research [37]. It is a Linux–based open-source software architecture which can be installed on any Linux operating systems—such as RHEL, Centos, Ubuntu, and Debian—without requiring any modification. Eucalyptus was adopted in this research because of the following advantages: (1) its compatibility with Amazon AWS APIs which means that Eucalyptus commands can be used to manage Amazon or Eucalyptus instances, and move freely between a Eucalyptus private cloud and the Amazon Public cloud, thus, making it a hybrid cloud; (2) its architecture is flexible enough to support businesses of any size and is highly scalable because of its distributed nature; (3) it allows in-house development of apps, which can then be migrated to the AWS. The main components of our Eucalyptus installation include:

(1) Cloud controller (CLC): This is known as an entry-point into the cloud for administrators, managers, developers, and end-users, and is responsible for meeting the requirements of node managers. Additionally, high-level scheduling decisions are made and implemented by the CLC with the help of cluster controllers.

(2) Cluster controller (CC): It is generally executed on a computer system that has network connectivity to systems running the Node Controllers (NCs), and to machines running the CLC. This component is responsible for managing a number of Virtual Machines (VMs) and scheduling the execution of VMs on particular NCs.

(3) Node controller (NC): It is executed on every system that is selected for hosting VM instances. The NC is responsible for interacting with the OS and the hypervisor running on the same system, and on the CC to take control of the life cycle of instances.

(4) Storage controller (SC): It implements block-accessed network storage such as EBS (Amazon Elastic Block Storage). Hence, the SC can send disk traffic across the local network to a remote storage site.

(5) Walrus: It allows different users to store their data systematically and define access control policies for authorized users to perform certain operations such as delete, and create. This component is actually a file-level storage system, but also represents a block-level storage system. Its interface is well-matched with Amazon’s S3 to store and access both the virtual machine images and user data.

Our experiment was implemented using C on a system with an Intel Core i5-2450 M CPU at 2.5 GHz, and 6 GB RAM. We also used the Pairing-Based Cryptography (PBC) version 0.5.14 to simulate the Wang and Yang’s data auditing schemes. We applied elliptic curve of 160-bit group order for both of the works. The results are obtained by the average of 20 trials.

The Algebraic signature is computed along with the groundwork of defining multiplication by using Galois’ theorem $GF(2^m)$ [43] as a polynomial multiplication modulo, where $g$ can be 16-bit (half-word) or 32-bit word. The addition of two polynomials is computed by the bitwise XOR of the string. Furthermore, the multiplication by the unknown $X$ is carried out by a left-shifting and making XOR with a parameter corresponding to the generator polynomial. Consequently, a $\gamma$ can be identified with the unknown, so that multiplication by $\gamma$ includes a left-shift operation followed by a conditional XOR. Broder [12] proposed a technique for performing several shift operations simultaneously, by creating a table which consists of a number of decisions that are used as the XOR-operand. The length of the signature indicates that the number of bits of algebraic signature is equal to 256 bits.

To demonstrate the efficiency of the proposed method in terms of frequent data update, as shown in Fig. 8, we conducted experiments for updating an outsourced 1 GB file, together with 125,000 data blocks where the number of updated blocks increases from 100 to 1000 with intervals of 8. To insert or delete a block ($i$) in the Wang scheme, the auditor has to find the precise position of the block ($i$) in the MHT tree and re-calculate the hash of the new leaf and existing nodes in the path to the root of the tree each time, and this incurs high computational overhead on the auditor. In the Yang method, after finding

<table>
<thead>
<tr>
<th>Metric</th>
<th>Scheme</th>
<th>[55]</th>
<th>[61]</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>Server</td>
<td>$O(\log m)$</td>
<td>$O(c)$</td>
<td>$O(c)$</td>
</tr>
<tr>
<td>Computation auditing</td>
<td>Server</td>
<td>$O(\log m)$</td>
<td>$O(cn)$</td>
<td>$O(cn)$</td>
</tr>
<tr>
<td>Computation modify</td>
<td>Verifier</td>
<td>$O(\log m)$</td>
<td>$O(c)$</td>
<td>$O(c)$</td>
</tr>
<tr>
<td>Computation insert</td>
<td>Verifier</td>
<td>$O(\log m)$</td>
<td>$O(m)$</td>
<td>$O(\frac{c}{m})$</td>
</tr>
<tr>
<td>Computation delete</td>
<td>Verifier</td>
<td>$O(\log m)$</td>
<td>$O(m)$</td>
<td>$O(\frac{c}{m})$</td>
</tr>
<tr>
<td>Computation append</td>
<td>Verifier</td>
<td>$O(\log m)$</td>
<td>$O(c)$</td>
<td>$O(c)$</td>
</tr>
</tbody>
</table>

To address this problem and improve the auditing scheme, we designed a new data structure (DCT) to reduce computational overhead.
the position of the block \(i\), the auditor is required to shift the remaining \((n-i)\) blocks for every insert or delete operation. This process, if repeated many times, will incur high computational overhead on the auditor. We have shown that our method reduces the computational cost on the auditor by using 10 DCTs with size 12,500 blocks instead of applying a single table with 125,000 blocks, as in the Yang scheme.

To investigate the impact of the number of divisions \(k\) on the computational cost of data update in the proposed method, we can define a scenario which involves different number of outsourced data blocks with size of 1 GB. Fig. 9 shows that by increasing the number of divisions \(k\) and approaching the optimum number \(\sqrt{125,000} \approx 352\) the computational cost on the auditor will decrease. However, when the number of updated blocks increases, the considerable computation overhead is incurred on the auditor. One of the best ways to reduce such high computational overhead is to use the optimum number of divisions based on the file size. For example, when the number of divisions is \(k = 100\), the computational time taken for inserting 1000 blocks is 0.156 s. Upon increasing the number of divisions to 353, the computational time falls to 0.140 s.

Fig. 10 shows the impact of file size on the computational cost of dynamic data update when the file size ranges from 1 GB to 10 GB, and 100 blocks are randomly inserted into or deleted from each of the outsourced files. When the size of the file increases to 10 GB, the computational time incurred using the Wang method increases markedly from 0.8 s to 2.3 s because of the need to manage a huge number of data blocks in the MHT. Similarly, by increasing the number of data blocks in the Yang scheme, the auditor has to shift a large number of blocks in order to insert or delete a data block that incurs high computational cost on the auditor. Fig. 10 shows that our method incurs minimum computational overhead on the auditor (maximum of 0.2 s for a 10 GB
file) because we use 10 DCTs instead of one large DCT, and the use of the algebraic signature. Therefore, our method is applicable for auditing large-scale files with dynamic properties.

Fig. 11 shows that when the size of the outsourced files or the number of data blocks increases, high computational overhead is incurred on the auditor. For example, the computational time for updating 1000 blocks of a file of 1 GB is 0.218 s, and when the file size increases to 100 GB, the overhead increases to 10.811 s (when the number of divisions is 10). The result also shows that the computational overhead falls markedly when the number of divisions is increased as the overhead for updating 1000 blocks of a file of 100 GB is only 0.296 s. Therefore, our method has a marked positive impact on the computational overhead when the data owner uploads a huge file or updates the outsourced file frequently.

8. Conclusions and future work

We present an efficient remote data auditing scheme to verify the integrity of the data stored in cloud computing. Our method employs the algebraic properties of the outsourced data blocks to remotely check the integrity of the files, and reduce the computational overhead on the client and server side of the cloud. We designed a new data structure—divide and conquer table—to support dynamic data update that incurs minimum computational and communication costs on the auditor. Using our DCT data structure, data owners can perform modify, delete, insert, or append operations at block level without having to download the whole file. Our DCT data structure can also be applied for large-scale data and will incur minimum computational cost on

Please cite this article as: M. Sookhak et al., Dynamic remote data auditing for securing big data storage in cloud computing, Information Sciences (2015), http://dx.doi.org/10.1016/j.ins.2015.09.004
the auditor and server. We assessed the security feature of our method by using a mathematical approach for validation. We compared our method with two state-of-the-art remote data auditing methods based on the computational cost involved in performing dynamic data update. The results show that our DCT data structure reduces the processing time of dynamic data update operations by decreasing the number of shifting. We also note that our DCT data structure markedly decreases the computational cost of dynamic data update for large-scale outsourced file in cloud computing.

We presented a single remote data auditing method for outsourced files in cloud computing that incurs minimum processing time and communication overhead. As part of future work, we will focus on extending our method for auditing the integrity of large archival files in distributed cloud storage systems.

Acknowledgments

This work is carried out as part of the Mobile Cloud Computing research project funded by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant, reference number UM.C/ HIR/ MOHE/FRGS/10, and reference number CR010-2014. We extend our sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding to Prolific Research Group (PRG-1436-16). This collaboration is partially supported by Melbourne-Chindia Cloud Computing (MC3) Research Network and Future Fellowship of Australian Research Council.

References


Please cite this article as: M. Sookhak et al., Dynamic remote data auditing for securing big data storage in cloud computing, Information Sciences (2015), http://dx.doi.org/10.1016/j.ins.2015.09.004