Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.016000

ARTICLE

A Deletable and Modifiable Blockchain Scheme Based on Record Verification Trees and the Multisignature Mechanism

Daojun Han[1,2,3], Jinyu Chen[3,4], Lei Zhang[1,2,3,*], Yatian Shen[1,2,3], Yihua Gao[3,5] and Xueheng Wang[3,6]

[1]Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng, 47500, China

[2]Institute of Data and Knowledge Engineering, Henan University, Kaifeng, 47500, China

[3]School of Computer and Information Engineering, Henan University, Kaifeng, 47500, China

[4]Department of Network and Finance, Agricultural Bank of China Limited Xuchang Branch, Xuchang, 461000, China

[5]School of Software, Pingdingshan University, Pingdingshan, 467000, China

[6]Department of Network and Finance, Agricultural Bank of China Limited Nanyang Branch, Nanyang, 473000, China

[*]Corresponding Author: Lei Zhang. Email: zhanglei@henu.edu.cn

Received: 30 January 2021 Accepted: 02 April 2021

ABSTRACT

As one of the most valuable technologies, blockchains have received extensive attention from researchers and industry circles and are widely applied in various scenarios. However, data on a blockchain cannot be deleted. As a result, it is impossible to clean invalid and sensitive data and correct erroneous data. This, to a certain extent, hinders the application of blockchains in supply chains and Internet of Things. To address this problem, this study presents a deletable and modifiable blockchain scheme (DMBlockChain) based on record verification trees (RVTrees) and the multisignature scheme. (1) In this scheme, an RVTree structure is designed and added to the block structure. The RVTree can not only ensure that a record is true and valid but, owing to its unique binary structure, also verify whether modification and deletion requests are valid. (2) In DMBlockChain, the multisignature mechanism is also introduced. This mechanism requires the stakeholders' signatures for each modification or deletion request and thus ensures that a record will not be modified arbitrarily. A user's request is deemed valid only if it is dually verified by the RVTree and the multisignature mechanism. The analysis finds that DMBlockChain can provide a secure and valid means for modifying and deleting records in a block while ensuring the integrity of the block and that DMBlockChain can effectively save space in some scenarios that require frequent records modification.

KEYWORDS

Blockchain; record verification trees; multisignature; DMBlockChain

1 Introduction

The concept of blockchains has received extensive attention from researchers since it was first put forward. Blockchains are widely applied in fields such as digital currencies [1], supply chains [2,3], the Internet of Things [4–7], intelligent healthcare [8–10], and intelligent agriculture [11–14]. Currently, blockchains are primarily used to store transactions and data. Available

blockchains meet storage needs for transactions. Blockchains can satisfy user requirements for data storage, such as decentralization, trust building, and data tamper-proofing, compared to conventional centralized storage. However, some researchers have found that the non-revisability of blockchain data can be very inconvenient in three main ways. First, Erroneous information cannot be modified [15,16]. Blockchains are used extensively to store certificates, health records, Internet of Things (IoT) data, and asset information. The authenticity and validity of these data are ensured through storage in blockchains. However, writing data into a blockchain as read-only prevents erroneous information in records from being corrected. Second, Sensitive information cannot be deleted. The basic data storage function of blockchains is often abused for storing pornography and illegally obtained data by violating intellectual property rights. For example, there are large stores of child pornography and illegal intellectual data in Bitcoin blockchains [17–21]. Third, Invalid data cannot be cleaned. After a blockchain has been in operation for a period of time, blocks generated early on may contain large quantities of undeletable redundant data [22]. These data take up a large amount of storage space, which wastes resources and may also violate relevant laws. For example, both the General Data Protection Regulation of the European Union and the Fair Credit Reporting Act of the United States consider the right to be forgotten a fundamental right of data subjects.

Thus, researchers have developed techniques to modify and delete block records, such as chameleon-hash functions and polynomials, that improve the hash verification structure and transaction mode of blocks. Most of these schemes realize the modification and deletion of individual data, i.e., a user can alter block data after passing node authentication, and other users cannot interfere in these alteration operations. However, this design is inapplicable to some scenarios. For example, in a supply chain, an enterprise may need to upload data to a blockchain for inspection by other enterprises or government agencies. The enterprise can modify erroneous data in the blockchain in a timely fashion to minimize effects on the normal production of other enterprises. However, the ability of an enterprise to freely modify uploaded data could impact governmental monitoring. For example, an enterprise could modify product data with quality issues to evade penalties. This problem also exists in other fields, such as the IoT and medical services. Therefore, a blockchain design that allows block data to be modified and deleted while simultaneously restricting user operation has relatively high practical significance for fields such as supply chains and the IoT.

1.1 Contribution

This paper presents a deletable and modifiable blockchain scheme (DMBlockChain) based on record verification trees (RVTrees) and the multisignature scheme. We make the following contributions.

(1) In this study, data modification and deletion functions are achieved at the transaction level. In our scheme, a record verification tree (RVTree) structure is designed and added to the block structure. A user can modify or delete data simply by changing the record list without modifying other data in the block. This operation does not affect the verification relationship in the block data, and other users can directly see the data modification history.

(2) In this study, a multisignature mechanism is introduced. A modification operation can pass node authentication only when a sufficient number of stakeholders sign the modification record. This restriction on the information owner protects the interests of stakeholders and makes malicious attacks more difficult.

(3) An experiment based on a consortium blockchain is performed in this study. The experimental results show that the proposed scheme realizes block data modification and deletion functions and effectively saves space in scenarios requiring frequent modifications and deletions.

1.2 Organization

The rest of the paper is organized as follows. Section 2 introduces the related work of this paper. Section 3 introduces relevant knowledge on bilinear mapping, the Lagrange interpolation method, and security models. Section 4 describes DMBlockChain and its block structure as well as how to modify and delete block records in detail. Section 5 presents an analysis of the integrity and security of DMBlockChain as well as an experimental analysis of its performance. The final section summarizes this paper and presents a preliminary discussion on noteworthy research directions.

2 Related Works

The tamper-proof property of conventional blockchains relies on integrity verification. Fig. 1 shows the two main processes of blockchain integrity verification within a block and for the entire block. The first process comprises integrity verification of the body and head of the block. A Merkle tree verifies the transaction authenticity of the block head (Fig. 1, relationship ①). The integrity of the Merkle tree is verified by the Merkle root of the block head (Fig. 1, relationship ②). The authenticity of the Merkle root of the block head is verified by the hash of the block head (Fig. 1, relationship ③). For an entire block, the head hash is verified by the subsequent block (Fig. 1, relationship ④). Blocks are connected by successive hash values into a chain. This chain structure ensures the integrity and tamper-proofness of the blocks.

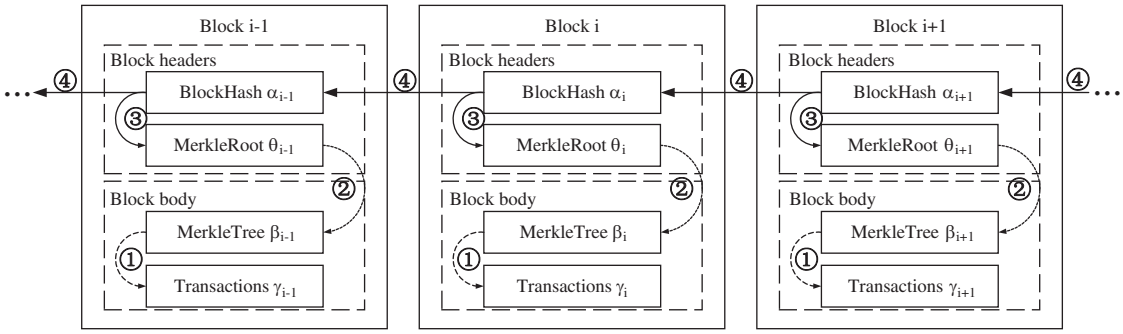

Figure 1: Verification relationship of blockchain

For blockchains, the block head hash and Merkle tree guarantee the integrity of the block horizontally and vertically, respectively. Therefore, the design of editable blockchains must start with the verification relationship, that is, implement the editing function of block data while ensuring the integrity of the verification relationship. The literature shows that editable blockchain schemes have been designed to begin with relationships ① and ④.

(1) Relationship ① refers to the verification relationship between the Merkle tree and the transaction, where in the Merkle tree guarantees the authenticity of the transaction by using a hash algorithm. Several researchers have started with this relationship by changing the verification relationship in block to enable the modification and deletion of block data. For example, Ren et al. [23] proposed a deletable blockchain scheme based on threshold ring signatures. In this

scheme, after receiving agreement and signatures from the majority of the nodes, the system can effectively delete the blockchain data while maintaining the overall blockchain structure. Later, Ren et al. [24] proposed a revisable blockchain scheme based on a trapdoor one-way function. In this scheme, a manoeuvring factor is introduced to reconstruct the signature subblock of each block. Block data can be validly modified as long as a specific threshold number of nodes agree. Additionally, the modification should not destroy the block-linking structure. The whole network can still verify data validity by the original verification procedure. Puddu et al. [17] proposed a variable blockchain scheme based on the PoW consensus protocol, whereby a specific type of transaction executes the modifications while the user monitors these changes. In addition, the scheme designed two new applications, namely, a collaborative recommendation system that can review inappropriate content and a time-locked encryption mechanism that can decrypt messages after a certain period of time. Lee et al. [18] implemented the modification and deletion of transactions by means of truncated hashes of the transaction content, and only the modified version is required to have a truncated hash equal to its original target value when a record must be modified.

(2) Relationship ④ refers to the verification relationship between blocks, which is crucial in implementing a chain structure. The current block header contains the block header hash of the previous block to ensure the integrity of the previous block. The integrity of the current block is ensured by the next block. Several researchers have started with this relationship by changing the verification relationship between blocks to enable the modification and deletion of block data. For example, Based on a chameleon hash, Accenture proposed a revisable blockchain scheme [25]. This scheme finds a collision of the existing data by finding the trapdoor of the chameleon hash and thereby revises the blockchain data. Subsequently, References [15,19,20,22] combined the secret sharing, trapdooring, and access structures to improve the chameleon hash in varying degrees on the basis of Reference [25] and enhance its performance and functional extensions. Deuber et al. [21] proposed the first editable decentralized blockchain protocol for permissionless systems based on the proof of work (PoW) consensus that can be easily adapted to any consensus mechanism. Cheng et al. [16] put forward a novel polynomial-based blockchain structure that organizes data segments in each block using Lagrange interpolation methods to enable block data editing.

These schemes can implement the modification and deletion of block [26] data but is only partially applicable to supply chains, Internet of Things (IoT) [27,28], and other scenarios because of the limitations of the consensus mechanism and scope of application as well as the unconstrained editing behavior of users [29,30].

3 Related Knowledge

To better describe DMBlockChain, the relevant knowledge used in this scheme is first presented.

3.1 Bilinear Mapping

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be p-order (p is a prime number) cyclic groups and g the generator of $\mathbb{G}_1$. Thus, a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties:

(1) Bilinearity. For $\forall x, y \in \mathbb{Z}_p$ and $\forall \partial,\ \beta \in \mathbb{G}_1,\ e(\partial^x, \beta^y) = e(\partial, \beta)^{xy}$.

(2) Non-degeneracy. When $\forall \partial, \beta \in \mathbb{G}_1,\ e(\partial, \beta) \neq 1_{\mathbb{G}_2}$.

(3) Computability. There is an effective algorithm for calculating $e(\partial, \beta)$ when $\forall \partial, \beta \in \mathbb{G}_1$.

3.2 Lagrange Interpolation Method

For an unknown n-order polynomial, if its (n + 1) number of points (x_i, y_i) are known, only one Lagrange interpolation polynomial $P_{(x)} = \sum_{j=0}^{n} y_j \Delta_j(x)$ can be determined, where $\Delta_j(x)$ is the interpolation basis function with the following expression:

$$\Delta_j(x) = \prod_{i=0, i\neq j}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{(x - x_{j-1})(x - x_{j+1})}{(x_j - x_{j-1})(x_j - x_{j+1})} \cdots \frac{x - x_n}{x_j - x_n} \tag{1}$$

3.3 Security Model

In this section, the security model used in this study is given to facilitate the subsequent security analysis of DMBlockChain. In classic mode, security is defined by a security game between a challenger and an adversary, which is defined as follows:

Setup: The adversary A gives a record verification tree *RVTree** to be challenged and sends it to the challenger C.

Phase 1. The adversary A produces keys corresponding to the record sets $S_1, S_2, \ldots, S_{q_1}$. None of $S_1, S_2, \ldots, S_{q_1}$ satisfy *RVTree**.

Challenge. The adversary A submits two random secret values $M_0, M_1 \in \mathbb{Z}_q$. The challenger C selects a random number $b \in \{0, 1\}$ and encrypts M_b in *RVTree**. The verification parameter *VP** is given to the adversary A.

Phase 2. The adversary A repeats Phase 1. However, none of the record sets $S_{q_1+1}, \ldots, S_q$ satisfy the *RVTree** corresponding to the challenge.

Guess. The adversary A outputs the guess b' of b.

In this game, the advantage of the adversary A is defined as follows:

$$Adv_A = \left| Pr\left[b' = b\right] - \frac{1}{2} \right| \tag{2}$$

Definition 1. If the advantage of the adversary within the polynomial time is negligible in the above game, then the scheme proposed in this study is considered secure.

4 DMBlockChain

To address the blockchain data modification and deletion problems, this study presents a feasible scheme, DMBlockChain, based on access tree-based secret sharing technology. DMBlockChain validly deletes and modifies the records in the blocks without damaging the integrity of the blockchain. DMBlockChain is characterized by decentralization, tamper-proofing, and valid modification and deletion. The conventional blockchain scheme is more similar to an account book, whereas DMBlockChain is more similar to a database system. The core of DMBlockChain lies in the introduction of an RVTree-based data structure. The verification mechanism of this structure allows a user to validly modify and delete the records in the blocks. A multisignature technology is also introduced. The stakeholders of a record are required to sign the record, which prevents the record owner from freely modifying or deleting the record. This section first introduces the RVTree design scheme and then describes the block structure and how blocks are modified and deleted.

4.1 RVTree

4.1.1 Overview

An RVTree is a binary and ternary mixed tree, as shown in Fig. 2. This tree contains a number n of layers. From the first layer to the $(n-2)^{th}$ layer are structure trees. Each structure tree is a ternary tree; i.e., each node of a structure tree has three leaf nodes. Each node of a structure tree represents a threshold gate. From the n^{th} layer to the $(n-1)^{th}$ layer are information trees. Each information tree is a binary tree; i.e., each node of an information tree has two leaf nodes.

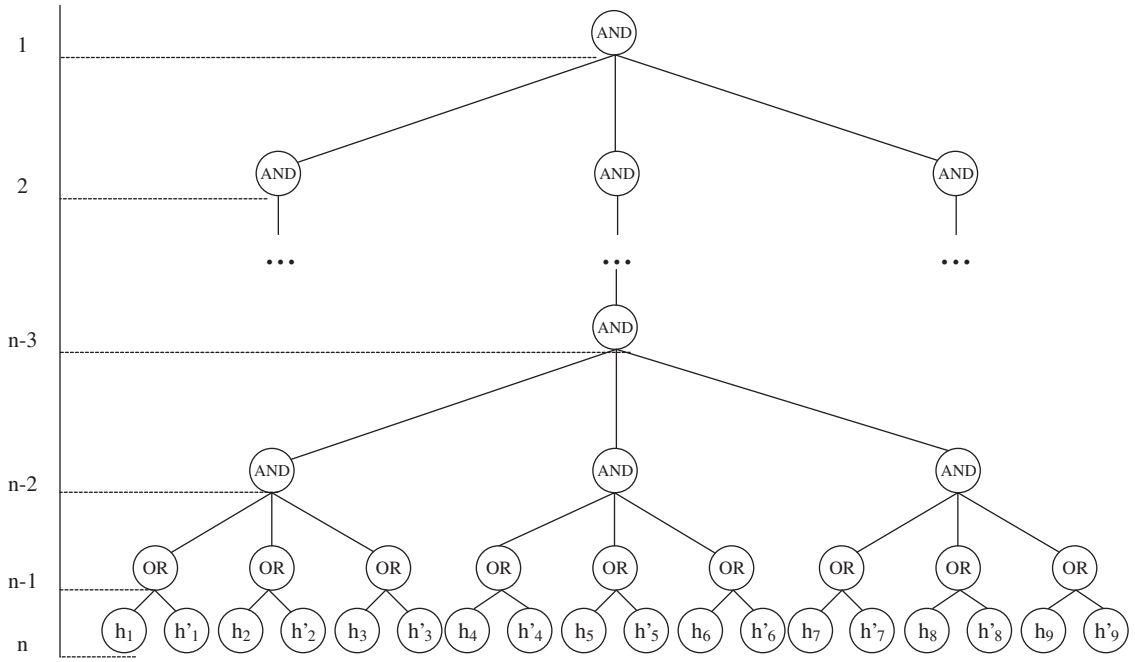

Figure 2: Record verification trees (full)

An RVTree is created by the following procedure.

First, the system establishes an information tree for each record (to verify authenticity). One leaf node stores the hash of the record (to verify the validity of the modification and deletion operations), and another leaf node stores the hash of the signature of the record. The parent node stores an OR gate. In our design, RVTree verifies not only the original record but also the modification or deletion operation and uses OR gate to connect the two hashes when one of the hashes can be verified by RVTree as a way to modify or delete the record.

Second, from the bottom up to the $(n-2)^{th}$ layer, every three information trees constitute one structure tree. The root node of each structure tree represents an AND gate. Starting from the $(n-3)^{th}$ layer, every three structure trees constitute one composite structure tree. The root node of each composite structure tree represents an AND gate. When there remains only one root node, an RVTree is established. AND gate are used in this design because each information tree represents a record, and we need to connect all the records in the block to form a whole and secure the records. Records are compared with one another, and AND gate is used to connect multiple records effectively. Each structure tree contains no more than three child nodes that

are not leaf child nodes. This design reduces the complexity in decrypting the secret value of *VP*. If one node is used to connect all the binary trees, the order of the polynomial could be exceedingly high. This could lead to an enormous computational load during the decryption process and thereby render it even more difficult to verify the integrity of the block. If less than three information or structure trees remain, then the remaining one or two will form a tree. The structure of a record verification tree with five records is illustrated in Fig. 3.

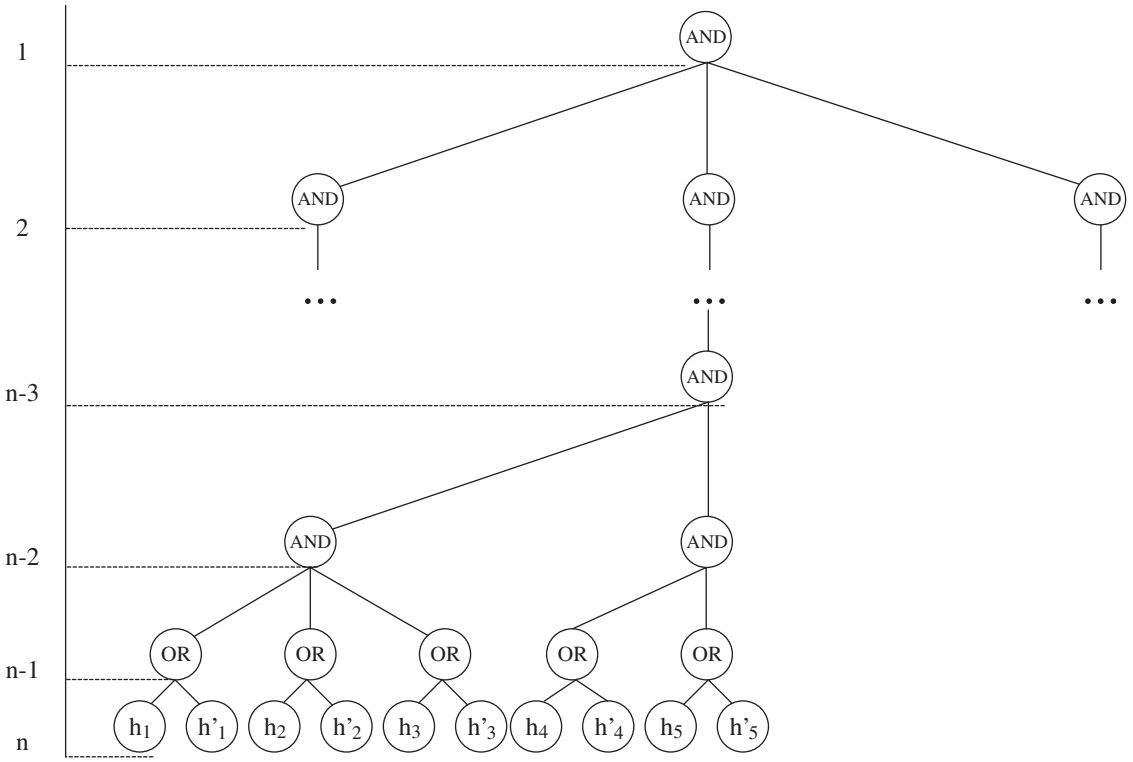

Figure 3: Record verification trees (non-full)

4.1.2 RVTree Scheme

There is a system of polynomials that corresponds to the RVTree. Here, the following functions are first defined. Index(x) represents the index of the node x. Parent(x) represents the parent node of the node x. The hash function H: $\{0,1\}^* \rightarrow \mathbb{G}_0$ maps any records in binary strings to random set elements. For a leaf node x, let $hash_x$ be the hash digest of the record related to x.

The initialization, generation, and verification algorithms are the main core RVTree algorithms. They are described below:

Setup. The initialization algorithm is run by the authorization centre. The initialization algorithm first selects a q-order bilinear group $\mathbb{G}_0$ with g as its generator and then generates a system security parameter SP.

$$SP = \{\mathbb{G}_0, g, e(g,g)\} \tag{3}$$

Generate (SP, RL). The generation algorithm is run by the blockchain nodes. This algorithm creates an RVTree $\mathcal{T}$ based on the record set RL and the creation rule and then selects a

$(K_x - 1)$-order polynomial q_x for each node x of the RVTree $\mathcal{T}$. Subsequently, this algorithm randomly selects an $s \in \mathbb{Z}_q$ as the secret value stored by the access tree $\mathcal{T}$ and stores it in the root node R of the access tree $\mathcal{T}$. Let $q_R(0) = s$. For other nodes, let $q_x(0) = q_{parent(x)}(index(x))$. In addition, let Y be the set of records stored in all the leaf nodes of the access tree $\mathcal{T}$. The hash of verification parameter *VPHash*, and the hash of the secret value *SHash* are calculated.

$$VP = \left\{ \mathcal{T}, SP,\ \forall y \in Y:\ C_y = g^{q_y(0)},\ \ C'_y = H\left(hash_y\right)^{q_y(0)} \right\} \tag{4}$$

$$VPHash = hash\left(VP\right) \tag{5}$$

$$SHash = hash\left(VPHash \cdot e\left(g, g\right)^{s}\right) \tag{6}$$

The blockchain nodes package *VPHash*, and *SHash* and other block information into a block and broadcast it to the whole network. After being verified, the block can be stored on the blockchain.

Verification (RL, VP). Verification algorithm. The key generation algorithm uses RL and VP as inputs and ultimately outputs the verification result.

First, the system executes a decryption algorithm *DecryptNode* (VP, x). This algorithm calculates the values stored in the nodes, starting from the leaf nodes of the access tree. In addition, this algorithm recursively calculates the values of the nodes of the previous layer. Through layer-by-layer calculations, this algorithm could ultimately decrypt the secret value of the root node. There are two scenarios in this process.

For the leaf nodes, let i be the value recorded in the node x.

If $i \notin RL$, then $DecryptNode\left(VP, x\right) = \perp$.

If $i \in RL$

$$DecryptNode\left(VP, x\right) = \frac{e\left(g \cdot H\left(i\right), g^{q_x(0)}\right)}{e\left(g, H\left(i\right)^{q_x(0)}\right)} = e\left(g, g\right)^{q_x(0)} \tag{7}$$

For the non-leaf nodes, decryption operations are performed layer by layer from top to bottom. For all the child nodes z belonging to the node x, the algorithm calls *DecryptNode* (VP, z) and stores the output as $F_z S_x$ is a set of a number k_x of nodes z in which $F_z \neq \perp$. If there is no such set, the function returns $\perp$. Otherwise, based on Eq. (1), calculate:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \quad where \begin{array}{l} i = index\left(z\right) \\ S'_x = \{index\left(z\right) : z \in S_x\} \end{array} \\ &= \prod_{z \in S_x} \left(e\left(g, g\right)^{q_z(0)}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} \left(e\left(g, g\right)^{q_{parent(z)}(index(z))}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} e\left(g, g\right)^{q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= e\left(g, g\right)^{q_x(0)} \end{aligned} \tag{8}$$

The algorithm then calculates the value of the root node R of the access tree.

$$A = DecryptNode\,(VP, z) = e\,(g, g)^{q_R(0)} = e\,(g, g)^{s} \tag{9}$$

Finally, let $Shash' = hash\,(A \cdot VPHash)$. Whether $Shash'$ is consistent with $Shash$ is determined. If $Shash'$ is consistent with $Shash$, the record is true and valid; otherwise, the record in the block has been tampered with.

4.2 Block Design

Fig. 4 shows the block design of DMBlockChain. In this scheme, the Merkle tree in a block is replaced by an RVTree (For details, see Section IV A). In addition, a verification parameter, a record list, and a new record list are added. A block is composed of a header and a body.

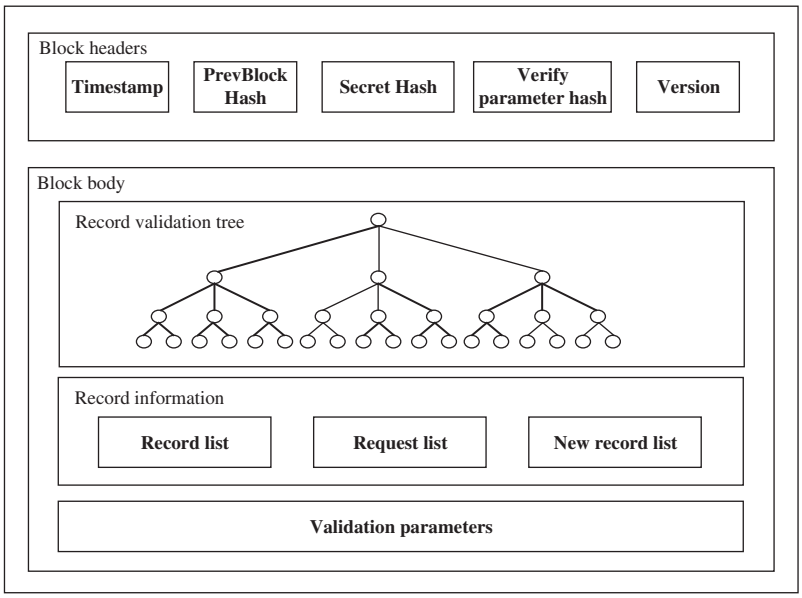

Figure 4: Block structure

Block header. The block header consists of the timestamp, the hash of the parent block, and the version number, which are in the original block structure, as well as a newly added the hash of verification parameter and a newly added the hash of the secret value. In the proposed scheme, the hash of the secret value is used to verify the authenticity of verification parameter. The hash of the secret value is calculated based on the secret value stored in the verification tree and the hash digest of verification parameter. The hash of the secret value is used to ensure the integrity of the record and the authenticity of verification parameter. The rest is consistent with the block structure of a conventional blockchain.

Block body: The block body consists of an RVTree, a record list, and a verification parameter. The RVTree is an important component that ensures the integrity of the block. The RVTree is connected upstream to the hash of the block and downstream from the record list to ensure the integrity of the record. The record list consists of a record list, a signature list, and a new record list. All records during the creation of the block are stored in the record list. The signature list and the new record list are in the default state. When a user deletes a record, the nodes delete

the record corresponding to the record list and then store the signature information relating to the user's deletion operation in the signature list. When a user modifies a record, the nodes also delete the record corresponding to the record list, then store the signature information relating to the user's modification operation in the record list, and finally store the updated record in the new record list.

4.3 Record Modification and Deletion

This section focuses on the description of the block modification and deletion processes. In addition, block generation is introduced. A comparison of block structure diagrams better shows the block modification and deletion processes.

4.3.1 Block Generation

The block generation process in the proposed scheme is similar to that in the conventional blockchain scheme. However, the main difference between the proposed and conventional blockchain schemes is that a Merkle tree is created in the conventional blockchain scheme, whereas an RVTree is created in the proposed scheme. Here, generation of three records is used as an example. Fig. 5 shows the created block structure.

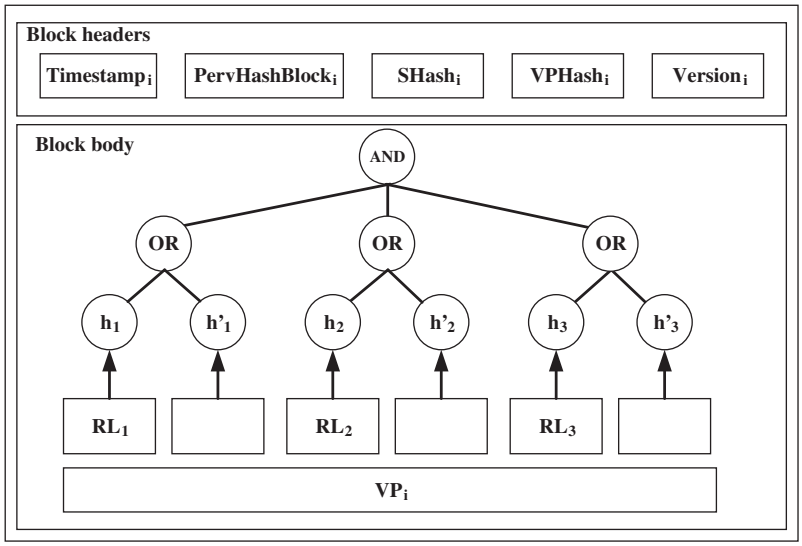

Figure 5: Block example (initial state)

4.3.2 Record Deletion

Fig. 6 is the flow chart of the record deletion process. Algorithm 1 shows the record deletion process. Fig. 7 shows the block structure after record deletion.

When a user wants to delete the record RL_1 in the block i, the user must first sign the hash $hash_1$ of RL_1 ($Sign(hash_1)$) and then generates a deletion request $DelInfo' = \{address, del, Sign(hash_1)\}$. In addition, the user sends the deletion request to the stakeholders. Stakeholders generally consist of business partners, industry associations, and government regulators. Each stakeholder signs the hash $hash_2$ of the deletion request. The file owner collects all the signatures and generates a message $DelInfo = \{address, del, Sign(hash_1), MuSign(hash_2)\}$ and broadcasts it to the blockchain network. All the nodes generate a hash digest $hash_{sign}$ by hashing

the signature information contained in the deletion request and subsequently substitute $hash_{sign}$, along with the hash set *hashlist* of the other records, into the RVTree algorithm *Verification*( ) and decrypt the secret value stored in the RVTree. Afterwards, the nodes obtain $hash_3$ by first calculating and then hashing the secret value and *VPHash*. The following verifications are then performed: (1) $hash_3$ is compared with *SHash*. (2) The multiple signatures are verified. If these two verifications are passed, then the deletion operation is accepted; otherwise, the deletion operation is rejected. Finally, the verification result is broadcast to the blockchain network. After receiving verification from the majority of the nodes of the network, the record RL_1 is deleted from the block, and the deletion request is inserted into the signature list.

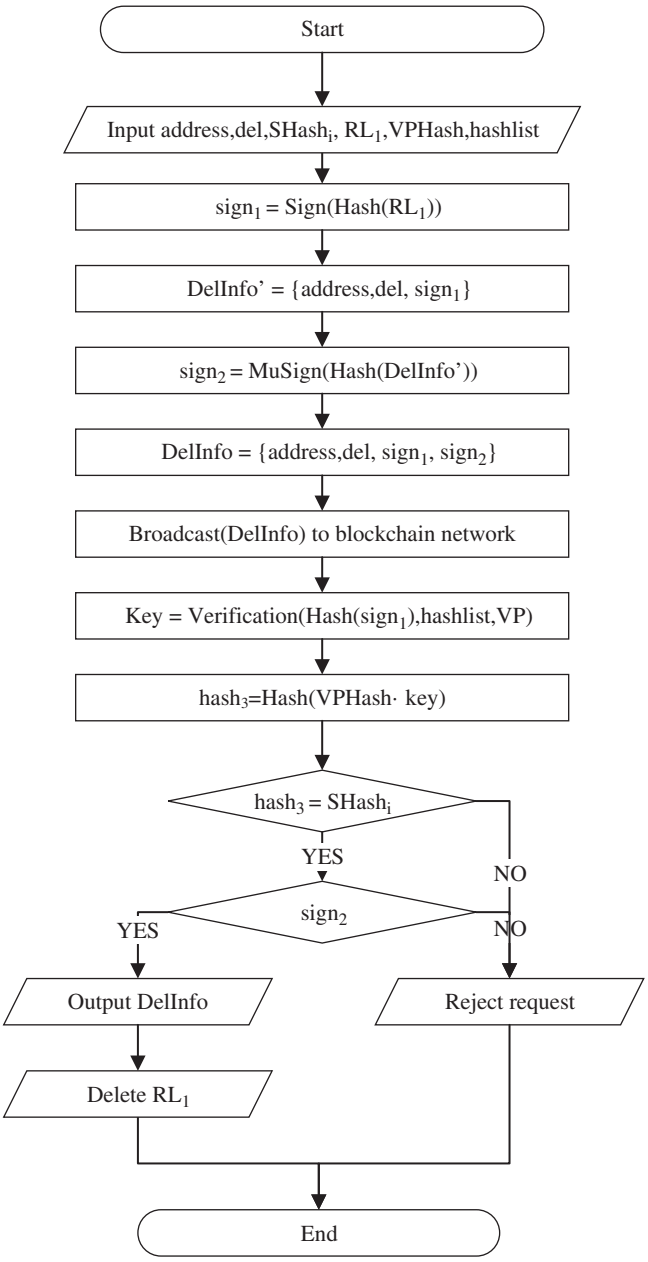

Figure 6: Flow chart of the record deletion process

Algorithm 1: An example of deleting block records

Input: RL_1: Record Information; *address*: Hash Address; *del*: Delete Sign; *Shash*: Secret Hash; *hashlist*: Record hash list; *VPHash*: Verify parameter hash;
Output: *DelInfo*: Delete Information

1. $hash_1 \leftarrow Hash\,(RL_1)$
2. $sign_1 \leftarrow Sign\,(hash_1)$
3. $DelInfo' \leftarrow \{address,\ del,\ sign_1\}$
4. $sign_2 \leftarrow MuSign\left(Hash\left(DelInfo'\right)\right)$
5. $DelInfo \leftarrow \{address,\ del,\ sign_1,\ sign_2\}$
6. $Broadcast\,(DelInfo)\ \ to\ blockchain\ network$
7. $hash_{sign_1} \leftarrow Hash\,(sign_1)$
8. $key \leftarrow Verification\left(hash_{sign_1},\ hashlist,\ VP\right)$
9. $hash_3 \leftarrow Hash\,(VPHash \cdot key)$
10. $if\ \ hash_3 = SHash$
11. $\quad then\ \ Result1 \leftarrow \{DelInfo,\ true\}$
12. $\quad else\ \ Result1 \leftarrow \{DelInfo,\ false\}$
13. $if\ \ sign_2$
14. $\quad then\ \ Result2 \leftarrow \{DelInfo,\ \ Yes\}$
15. $\quad else\ \ Result2 \leftarrow \{DelInfo,\ \ No\}$
16. $Result \leftarrow Result1 + Result2$
17. $Broadcast\,(Result)\ to\ blockchain\ network$
18. $if\ \ Result = (DelInfo, true, Yes)$
19. $\quad then\ \ Add\,(DelInfo)\ \ to\ block\ and\ \ Delete\,(RL_1)$
20. $\quad else\ \ Reject\ the\ request$

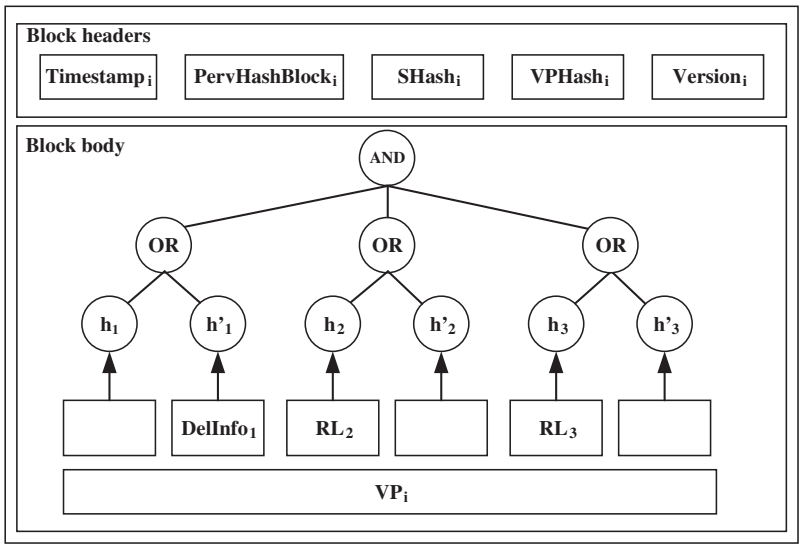

Figure 7: Block example (status after deletion of record)

4.3.3 Record Modification

Fig. 8 is the flow chart of the record modification process. Algorithm 2 shows the record modification process. Fig. 9 shows the block structure after record modification.

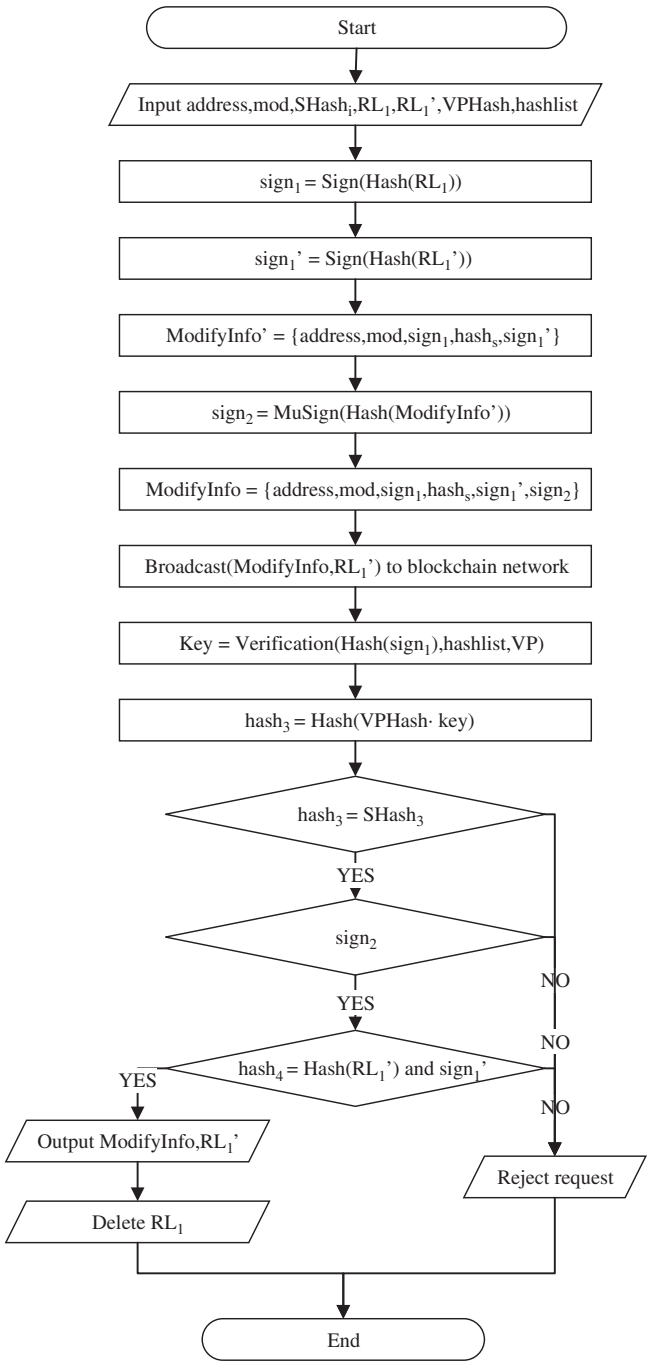

Figure 8: Flow chart of the record modification process

Algorithm 2: An example of modifying block records

Input: RL_1: Record Information; *address*: Hash Address; *mod*: Modify Sign; *Shash*: Secret Hash; RL_1': New Record; *hashlist*: Record hash list; *VPHash*: Verify parameter hash;
Output: *ModifyInfo*: Modify Information and RL_1'

1. $hash_1 \leftarrow Hash\,(RL_1)$
2. $hash_1' \leftarrow Hash\left(RL_1'\right)$
3. $sign_1 \leftarrow Sign\,(hash_1)$
4. $sign_1' \leftarrow Sign\left(hash_1'\right)$
5. $ModifyInfo' \leftarrow \left\{address,\ mod,\ sign_1,\ hash_1',\ sign_1'\right\}$
6. $sign_2 \leftarrow MuSign\left(Hash\left(ModifyInfo'\right)\right)$
7. $ModifyInfo \leftarrow \left\{address,\ mod,\ sign_1,\ hash_1',\ sign_1',\ sign_2\right\}$
8. $Broadcast\left(ModifyInfo,\ RL_1'\right)$ *to blockchain network*
9. $hash_{sign_1} \leftarrow Hash\,(sign_1)$
10. $key \leftarrow V\text{erification}\left(hash_{sign_1}, hashlist, VP\right)$
11. $hash_3 \leftarrow Hash\,(VPHash \cdot key)$
12. *if* $hash_3 = SHash$
13. *then* $Result1 \leftarrow \{ModifyInfo,\ true\}$
14. *else* $Result1 \leftarrow \{ModifyInfo,\ false\}$
15. *if* $sign_2$
16. *then* $Result2 \leftarrow \{ModifyInfo,\ Yes\}$
17. *else* $Result2 \leftarrow \{ModifyInfo,\ No\}$
18. $hash_4 \leftarrow Hash\left(RL_1'\right)$
19. *if* $hash_4 = hash_1'$ *and* $sign_1'$
20. *then* $Result3 \leftarrow \{ModifyInfo, Yes\}$
21. *else* $Result3 \leftarrow \{ModifyInfo, No\}$
22. $Result \leftarrow Result1 + Result2 + Result3$
23. $Broadcast\,(Result)$ *to blockchain network*
24. *if* $Result = (ModifyInfo,\ true,\ Yes,\ Yes)$
25. *then* $Add\left(ModifyInfo,\ RL_1'\right)$ *to block*
26. $Delete\,(RL_1)$
27. *else Reject the request*

When a user wants to modify the record RL_1 in the block i, the user must first sign the hash $hash_1$ of $RL_1 Sign\,(hash_1)$ and then obtains $hash_1'$ by hashing the new record RL_1'. In addition, the user must sign $hash_1'$ $Sign\left(hash_1'\right)$. Subsequently, the system generates a modification request $ModifyInfo' = \{address,\ mod,\ Sign\,(hash_1),\ hash_1',\ Sign\left(hash_1'\right)\}$. The user sends the modification request to the stakeholders. Each stakeholder signs the hash hash2 of the modification request. The file owner collects all the signatures, generates a message $ModifyInfo = \left\{address,\ mod,\ Sign\,(hash_1),\ hash_1',\ Sign\left(hash_1'\right)\right\}$, and sends it, along with the recreated record $RL_1{}'$, to the blockchain network. All nodes generate a hash digest $hash_{sign}$ by hashing the signature information contained in the modification request and subsequently substitute it, along with the hash set *hashlist* of the other records, into the RVTree algorithm *Verification*( ) and decrypt the secret value stored in the RVTree. Afterwards, the nodes obtain $hash_3$ by first calculating and then hashing the secret value and the hash of verification parameter. The following verifications are then performed: (1) $hash_3$ is compared with *SHash*. (2) The newly generated record is hashed,

and its hash is compared with the hash contained in the modification request. (3) The multiple signatures are verified. If these three verifications are passed, the modification operation is accepted; otherwise, the modification operation is rejected. Finally, the verification result is broadcast to the blockchain network. After receiving verification from the majority of the nodes of the network, the record RL_1 is deleted from the block, and the modification request and the new record are inserted into the signature list and the new record list, respectively.

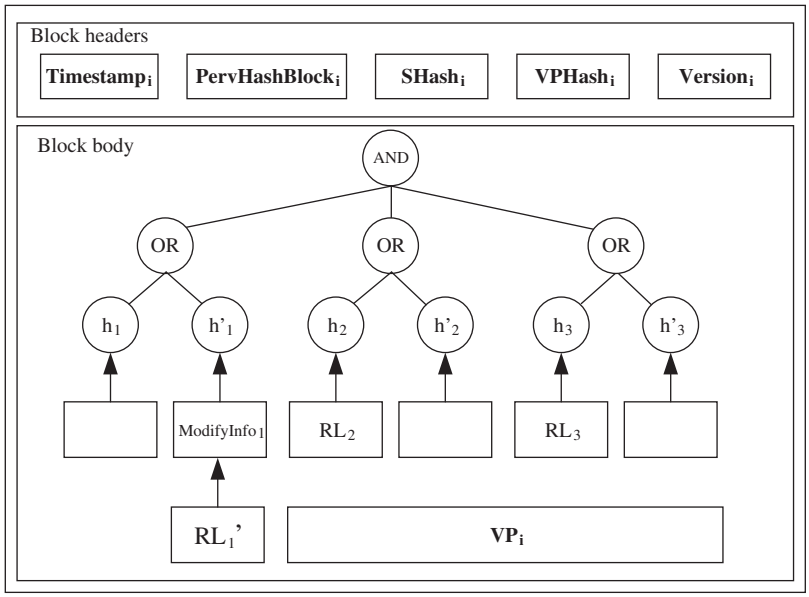

Figure 9: Block example (status after modifying records)

5 Analysis

The following analysis demonstrates the practical significance, effectiveness, feasibility, and security of the proposed scheme. First, DMBlockChain is compared with references [15–24]. Then, the security of DMBlockChain is analysed. Finally, the performance of DMBlockChain is analysed through experiments.

5.1 Comparative Analysis

In this section, DMBlockChain is compared with references [15–24]. As demonstrated in Tab. 1, in terms of scope of application, the scheme in references [15,17,18,21–24] is only applicable to one type of blockchain (e.g., PoW consensus-based blockchain), while DMBlockChain and that in references [16,19,20] are applicable to most types of blockchains with a broad scope of application. From the editing level, references [15,16,19–23] implemented only block-level data modifications, while DMBlockChain and references [17,18,24] performed record-level data modifications. In addition, DMBlockChain constrains the user's actions and is suitable for use in scenarios, such as supply chain and IoT.

Table 1: Comparison of schemes

	Deleteable	Modifiable	Key technology	Scope of application	Time complexity	Editorial level
DMBlockChain	Yes	Yes	RVTrees	Public blockchains, private blockchains, consortium blockchains	O(n)	Record
Traditional blockchain	No	No		Public blockchains, private blockchains, consortium blockchains		
[15]	Yes	Yes	Chameleon hash functions and secret sharing	Public blockchains, private blockchains, consortium blockchains	O(1)	Block
[19]	Yes	Yes	Chameleon hash functions	Public blockchains, private blockchains, consortium blockchains	O(1)	Block
[20]	Yes	Yes	Strategy-based chameleon hash function	Public blockchains, private blockchains, consortium blockchains	O(n)	Block
[22]	Yes	Yes	Chameleon hash functions	Public blockchains, private blockchains, consortium blockchains	O(n)	Block
[23]	Yes	No	Threshold ring signatures	Blockchains based on PoSpace	$O(n^2)$	Block
[24]	No	Yes	Trapdoor one-way function	Blockchains based on PoSpace	O(n)	Record
[17]	Yes	Yes	New transaction mechanism	Blockchains based on PoW	O(n)	Record
[18]	Yes	Yes	Truncated hash values	Blockchains based on PoW	O(n)	Record
[21]	Yes	Yes	Extended Protocol	Blockchains based on PoW	O(n)	Block
[16]	Yes	Yes	Polynomial function	Public blockchains, private blockchains, consortium blockchains	O(n)	Block

5.2 Security Analysis

Security analysis will be conducted on the solution of this study in terms of both block generation and record editing. Security analysis of the block generation process includes both security of the record verification tree and block integrity. The security of the block record editing process includes both block security during modification or deletion and interrupted operation security.

5.2.1 RVTree Security Analysis

Theorem 1. In a bilinear group model, the RVTree scheme proposed in this study is secure against chosen-plaintext attacks.

Proof: It is assumed that there is one adversary A that is able to break the proposed scheme. C is a challenger. The following operations are performed:

Setup. The adversary A gives a record verification tree $RVTree^*$ to be challenged and sends it to the challenger C.

Phase 1. The adversary A inquires the challenger C about the keys corresponding to the hash sets of multiple records, $S_1, S_2, \ldots, S_{q_1}$. None of $S_1, S_2, \ldots, S_{q_1}$ satisfy the $RVTree^*$.

Challenge. The adversary A submits two random secret values $M_0, M_1 \in \mathbb{Z}_q$. The challenger C selects a random number $b \in \{0, 1\}$. Let $q_R(0) = M_b$, which corresponds to the root node R of the RVTree $\mathcal{T}$, be equal to Mb. For other nodes, let $q_x(0) = q_{parent(x)}(index(x))$. Calculate:

$$VP = \left\{ \mathcal{T}, SP, \forall y \in Y: C_y = g^{q_y(0)}, C'_y = H\left(hash_y\right)^{q_y(0)} \right\} \tag{10}$$

The VP is then sent to the adversary A.

Phase 2. Similar to Phase 1, the adversary A continues to inquire the challenger C about the keys corresponding to $S_{q_1+1}, \ldots, S_q$. It is required that the hash sets of records, $S_{q_1+1}, \ldots, S_q$ do not satisfy the challenged $RVTree^*$.

Guess. The adversary A outputs the guess b' of b. If $b' = b$, when the adversary A decrypts the secret value, his or her advantage is equal to $Pr\left[b' = b\right] = \frac{1}{2} + \epsilon$. The advantage of the adversary within the polynomial time is negligible. The scheme established in this study is secure under the defined security model.

5.2.2 Block Integrity Analysis

For a conventional blockchain, the header of a block contains the hash of the previous block, which is obtained by hashing the header of the previous block. The blocks are linked by hashes successively to form a blockchain. This chain structure ensures the integrity and tamper-proofing of the blocks. Transaction authenticity is guaranteed by a Merkle tree. The hash of a block ensures its transverse integrity, whereas the Merkle tree ensures its longitudinal integrity.

In DMBlockChain, the Merkle tree is replaced by a record verification tree. The transverse integrity of the blocks relies completely on the chain structure formed by the hash of the block. The longitudinal integrity of the block relies completely on the record verification tree and verification parameter. Fig. 10 shows the correlations among the hash of the block, the hash of the secret value, record verification tree, the hash of verification parameter, and verification parameter. the hash of the block ensures the integrity of the chain structure between blocks and the authenticity of the hash of the secret value. The hash of the secret value ensures the authenticity of the record verification tree and the hash of verification parameter. The record verification tree ensures the authenticity of the record list. the hash of verification parameter ensures the authenticity of verification parameter.

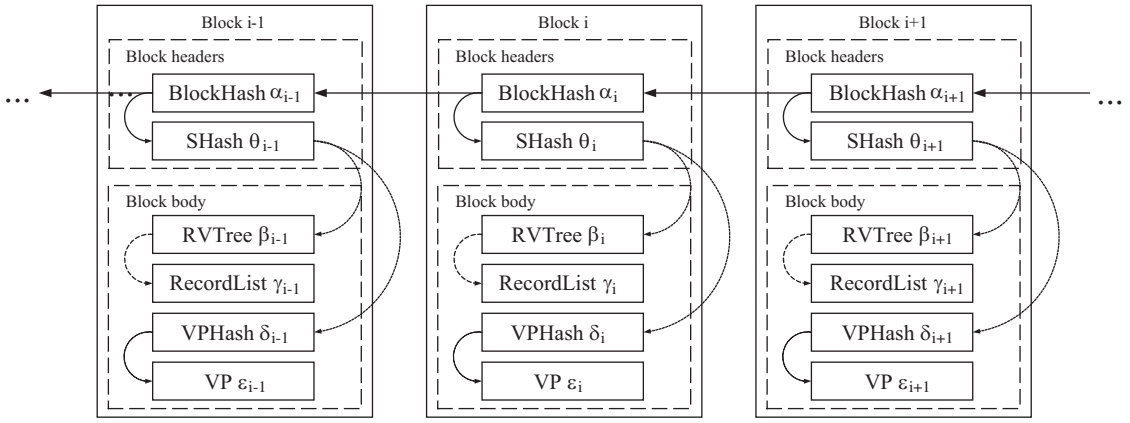

Figure 10: Block association

5.2.3 Security Analysis of the Blockchain at Record Modification or Deletion

During modification and deletion operations, only the recorded information is modified while the rest remains unchanged and the binding relationship between blocks is unaffected. If the block data of a single or multiple nodes is erroneous or lost, then these nodes will request block data from other nodes. When the lost data are received, the nodes will verify the authenticity of the data and add it to the block after passing verification.

There is a change in the data in a block after a record it contains is modified or deleted. Therefore, it is necessary to re-verify the integrity of the blockchain. Whether a valid verification can be established between the user's request and the block is the key to verification. An RVTree is the core of establishing a verification between the block and the user's request. In addition, the validity of the new record is guaranteed by the user's request. Therefore, it is necessary only to verify whether the new record and the modification or deletion request are valid.

Request security. The validity of a modification or deletion request is guaranteed by the RVTree. A leaf node of the RVTree stores the hash of the signature corresponding to the original record. The unidirectionality of the hash function ensures that the signature information for a user's record could not be falsified. In addition, the signature algorithm, to a certain extent, ensures the security of signature information. As long as owners keep their private keys securely, the forgery of the valid proof is impossible. Therefore, the authenticity of a user's request is considered secure as long as the request contains the correct signature information for the record.

New record security. When a user must modify a record, the user should submit a new record. The owner of the record should sign the hash of the record. The signature of the record owner ensures the validity of the new record. In addition, the stakeholders sign the whole modification request. The signature mechanism ensures the security of a new record.

In addition, the multisignature mechanism is introduced. The owner of a record can validly modify or delete the record only if the owner receives signatures from multiple stakeholders. Such a design constrains not only the user's actions but also increases the difficulty for malicious users to forge requests, thereby enhancing the scheme's resilience to attacks.

In summary, the RVTree ensures the validity of the user's request. The user's request ensures the validity of the new record. The hash, signature, and multisignature algorithms collectively ensure the security of the block at record modification or deletion.

5.2.4 Impact of Interrupt Operation

During the execution of a modification or deletion operation, the user may interrupt the operation, but the integrity of the block remains unaffected by this interruption. We analyze the operation according to the three stages of execution.

(1) From the user-generated request to the stakeholder signature stage. If a request is interrupted with a wish, then the operation is simply undone and the blockchain remains unaffected.

(2) From the user request broadcast in the blockchain network to the node verification request stage. In this phase, some nodes may have completed the requested operation. If a user requests an interrupt operation, then the node performing the completed operation validates the revocation

request. If the revocation is valid, then the node requests the original record from the other nodes and restores the original record. Nodes that do not perform operations abort them.

(3) The nodes are consistent, and the operation execution phase is complete. At this stage, the operation is complete and the user cannot interrupt it.

5.3 Performance Analysis

To test the feasibility and performance of DMBlockChain, simulations were performed in a hardware environment consisting of 16 computers, each with a 2.4-GHz Intel Core i7 processor and 4 GB of random access memory, and a software environment consisting of the Ubuntu 18 operating system, Python 3, and a consortium blockchain based on the Practical Byzantine Fault Tolerance consensus protocol. First, the block packaging times corresponding to various numbers of records were statistically analysed. The times required by the generation and verification processes of the RVTree and the Merkle tree were then statistically analysed. Finally, the conventional and proposed blockchain schemes were compared in terms of storage space occupancy. The experimental results showed that DMBlockChain not only realizes modification and deletion of block data but also ensures the integrity of block data.

Compared to the conventional blockchain scheme, DMBlockChain needs a relatively long time to generate an RVTree. Thus, the focus was placed on comparing the generation times for the RVTree and Merkle tree. In addition, the verification times for the RVTree and the Merkle tree were compared. Fig. 11 shows the statistics of the block packaging times of the system corresponding to various numbers of records. As demonstrated in Fig. 11, the time required by the system to package records was in direct proportion to the number of records.

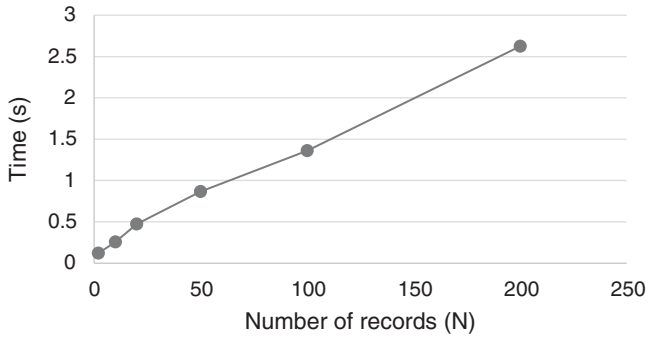

Figure 11: Record packaging efficiency

The RVTree and Merkle tree generation and verification processes were compared in terms of time efficiency. Fig. 12 shows the statistics of the generation times corresponding to various numbers of records. As demonstrated in Fig. 12, the time efficiency for Merkle tree generation and verification was consistent and increased relatively slowly. In comparison, in DMBlockChain, the RVTree generation and verification times increased rapidly with increasing number of records. Compared to the Merkle tree, it took a longer time to generate and verify an RVTree.

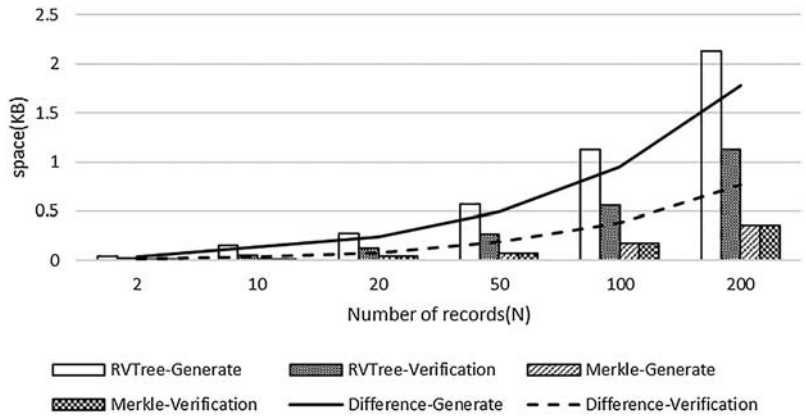

Figure 12: RVTree and Merkle generation and verification time comparison

Moreover, the RVTree and the Merkle tree were compared in terms of the space occupied by the generated data. Fig. 13 shows the generated data sizes corresponding to various numbers of records. As demonstrated in Fig. 13, there was a relatively insignificant change in the space occupied by the data generated by the Merkle tree. In comparison, in DMBlockChain, the space occupied by the RVTree increased rapidly with increasing number of records. Compared to the Merkle tree, the RVTree occupied a larger storage space. Furthermore, a block with 50 records was statistically analysed. On average, each record in the RVTree occupied a space of 1.4 K. By contrast, each record in the Merkle tree occupied a space of 0.02 K. Let n be the number of the modifications of the block and y the space saving rate; thus, $y = \frac{0.02n - 1.4}{0.02n} = 1 - \frac{70}{n}$ (Fig. 14). As demonstrated in Fig. 14, when n > 70, y > 0 for the RVTree. Therefore, it can be inferred that DMBlockChain can effectively save space in scenarios that require frequent record modification.

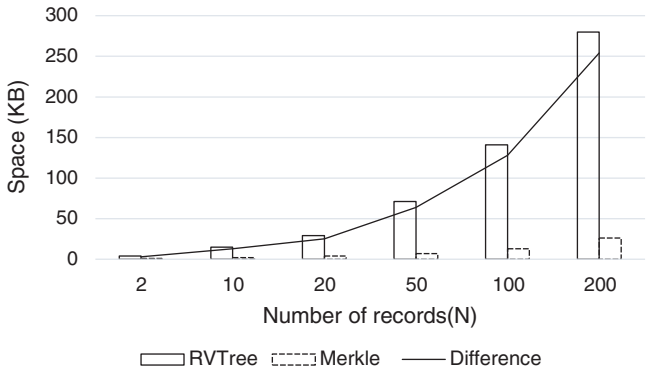

Figure 13: Storage space comparison between RVTree and Merkle

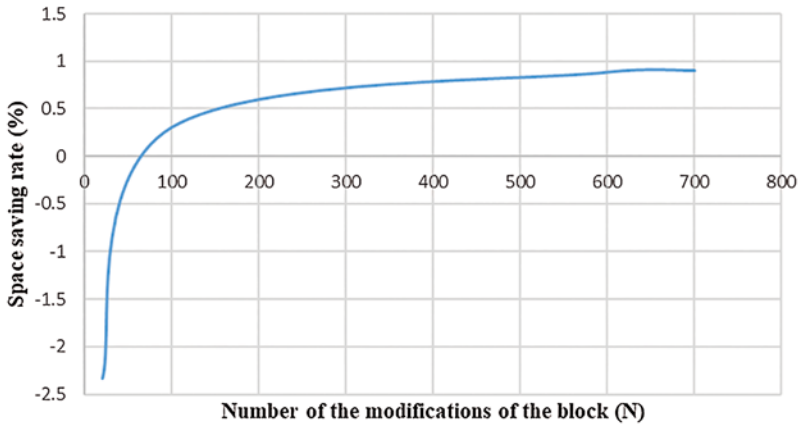

Figure 14: The change of occupancy of storage space

6 Conclusions

This study presents a blockchain scheme based on RVTrees and the multisignature mechanism DMBlockChain. In this scheme, an RVTree is introduced to the block structure, thereby realizing modification and deletion of records on the blockchain. The simultaneous introduction of multiple signature mechanisms to constrain the user's editing behavior is suitable for IoT and supply chain applications. Experimental simulation results show that DMBlockChain can realize deletion and modification of records in a block while ensuring its integrity. However, the space occupancy of DMBlockChain is relatively high, whereas its time efficiency is relatively low. Therefore, on this basis, future studies should focus on further reducing the generation time and space occupancy of the RVTree.

Funding Statement: This work was supported by the Scientific and technological project of Henan Province (Grant Nos. 202102310340, 212102210414), Foundation of University Young Key Teacher of Henan Province (Grant Nos. 2019GGJS040, 2020GGJS027), Key scientific research projects of colleges and universities in Henan Province (Grant No. 21A110005), and National Natual Science Foundation of China (61701170).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Hu, Y., Manzoor, A., Ekparinya, P., Liyanage, M., Thilakarathna, K. et al. (2019). A delay-tolerant payment scheme based on the ethereum blockchain. *IEEE Access, 7,* 33159–33172. DOI 10.1109/ACCESS.2019.2903271.
2. Islam, M. N., Kundu, S. (2019). Enabling IC traceability via blockchain pegged to embedded PUF. *ACM Transactions on Design Automation of Electronic Systems*, *24(3),* 1–23. DOI 10.1145/3315669.
3. Xu, X., Rahman, F., Shakya, B., Vassilev, A., Forte, D. et al. (2019). Electronics supply chain integrity enabled by blockchain. *ACM Transactions on Design Automation of Electronic Systems, 24(3),* 1–25. DOI 10.1145/3315571.
4. Christidis, K., Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. *IEEE Access, 4,* 2292–2303. DOI 10.1109/ACCESS.2016.2566339.

5. Sharma, P. K., Chen, M. Y., Park, J. H. (2017). A software defined fog node based distributed blockchain cloud architecture for IoT. *IEEE Access, 6,* 115–124. DOI 10.1109/ACCESS.2017.2757955.
6. Rahman, M. A., Rashid, M. M., Hossain, M. S., Hassanain, E., Alhamid, M. F. et al. (2019). Blockchain and IoT-based cognitive edge framework for sharing economy services in a smart city. *IEEE Access, 7,* 18611–18621. DOI 10.1109/ACCESS.2019.2896065.
7. Tsang, Y. P., Choy, K. L., Wu, C. H., Ho, G. T. S., Lam, H. Y. (2019). Blockchain-driven IoT for food traceability with an integrated consensus mechanism. *IEEE Access, 7,* 129000–129017. DOI 10.1109/ACCESS.2019.2940227.
8. Yang, X., Li, T., Pei, X., Wen, L., Wang, C. (2020). Medical data sharing scheme based on attribute cryptosystem and blockchain technology. *IEEE Access, 8,* 45468–45476. DOI 10.1109/ACCESS.2020.2976894.
9. Nguyen, D. C., Pathirana, P. N., Ding, M., Seneviratne, A. (2019). Blockchain for secure EHRS sharing of mobile cloud based e-health systems. *IEEE Access, 7,* 66792–66806. DOI 10.1109/ACCESS.2019.2917555.
10. Rahman, M. A., Hossain, M. S., Loukas, G., Hassanain, E., Rahman, S. S. (2018). Blockchain-based mobile edge computing framework for secure therapy applications. *IEEE Access, 6,* 72469–72478. DOI 10.1109/ACCESS.2018.2881246.
11. Zhang, S., Ye, J., Li, G. G. (2020). Research and implementation of blockchain technology scheme for cold chain logistics. *Computer Engineering and Applications, 56(3),* 19–27. DOI 10.3778/j.issn.1002-8331.1908-0453.
12. Mezquita, Y., González-Briones, A., Casado-Vara, R., Chamoso, P., Prieto, J. et al. (2019). Blockchain-based architecture: A MAS proposal for efficient agri-food supply chains. *International Symposium on Ambient Intelligence*, pp. 89–96. Cham: Springer.
13. Devi, M. S., Suguna, R., Joshi, A. S., Bagate, R. A. (2019). Design of IoT blockchain based smart agriculture for enlightening safety and security. *International Conference on Emerging Technologies in Computer Engineering*, pp. 7–19. Singapore: Springer.
14. Keke, W., Zhide, W., Jian, X. (2019). Efficient traceability system for quality and safety of agricultural products based on consortium blockchain. *Journal of Computer Applications, 39(8),* 2438–2443. DOI 10.11772/j.issn.1001-9081.2019020235.
15. Li, P. L., Xu, H. X., Ma, T. J., Mu, Y. H. (2018). Research on fault-correcting blockchain technology. *Journal of Cryptologic Research, 5(5),* 501–509. DOI 10.13868/j.cnki.jcr.000259.
16. Cheng, L., Liu, J., Su, C., Liang, K., Xu, G. et al. (2019). Polynomial-based modifiable blockchain structure for removing fraud transactions. *Future Generation Computer Systems, 99(11),* 154–163. DOI 10.1016/j.future.2019.04.028.
17. Puddu, I., Dmitrienko, A., Capkun, S. (2017). μchain: How to forget without hard forks. IACR Cryptology ePrint Archive. 2017:106.
18. Lee, N. Y., Yang, J., Onik, M. M. H., Kim, C. S. (2019). Modifiable public blockchains using truncated hashing and sidechains. *IEEE Access, 7,* 173571–173582. DOI 10.1109/ACCESS.2019.2956628.
19. Ateniese, G., Magri, B., Venturi, D., Andrade, E. (2017). Redactable blockchain-or–rewriting history in bitcoin and friends. *IEEE European Symposium on Security and Privacy*, pp. 111–126. Paris, France, IEEE. DOI 10.1109/EuroSP.2017.37.
20. Derler, D., Samelin, K., Slamanig, D., Striecks, C. (2019). Fine-grained and controlled rewriting in blockchains: Chameleon-Hashing gone attribute-based. *26th Annual Network and Distributed System Security Symposium*. DOI 10.14722/ndss.2019.23066.
21. Deuber, D., Magri, B., Thyagarajan, S. A. K. (2019). Redactable blockchain in the permissionless setting. *IEEE Symposium on Security and Privacy*, pp. 124–138. San Francisco, CA, USA, IEEE. DOI 10.1109/SP.2019.00039.
22. Rajasekhar, K., Yalavarthy, S. H., Mullapudi, S., Gowtham, M. (2018). Redactable blockchain and it's implementation in bitcoin. *International Journal of Engineering & Technology, 7(1),* 401–405. DOI 10.14419/ijet.v7i1.1.9861.
23. Ren, Y. N., Xu, D. T., Zhang, X. P., Gu, D. W. (2019). Deletable blockchain based on threshold ring signature. *Journal on Communications, 40(4),* 71–82. DOI 10.11959/j.issn.1000-436x.2019084.

24. Ren, Y. L., Xu, D. T., Zhang, X. P., Gu, D. W. (2020). A scheme of revisable blockchain. *Ruan Jian Xue Bao/Journal of Software, 31(12),* 3909–3922 (in Chinese).
25. Krawczyk, H. M., Rabin, T. D. (1998). Chameleon hashing and signatures. Patent 6,108,783.
26. Tian, Z., Li, M., Qiu, M., Sun, Y., Su, S. (2019). Block-DEF: A secure digital evidence framework using blockchain. *Information Sciences, 491(2019),* 151–165.
27. Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S. et al. (2020). A survey on access control in the age of internet of things. *IEEE Internet of Things Journal, 7(6),* 4682–4696. DOI 10.1109/JIOT.2020.2969326.
28. Luo, C., Tan, Z., Min, G., Gan, J., Shi, W. et al. (2021). A novel web attack detection system for internet of things via ensemble classification. *IEEE Transactions on Industrial Informatics,* DOI 10.1109/TII.2020.3038761.
29. Ma, Y., Wu, Y., Ge, J. (2020). *Accountability and privacy in network security*. Singapore: Springer.
30. Shafiq, M., Tian, Z., Bashir, A. K., Du, X., Guizani, M. (2020). CorrAUC: A malicious bot-IoT traffic detection method in IoT network using machine learning techniques. *IEEE Internet of Things Journal.* DOI 10.1109/JIOT.2020.3002255.