Toward Materials Genome Big-Data: A Blockchain-Based Secure Storage and Efficient Retrieval Method

Ran Wang, *Graduate Student Member, IEEE*, Cheng Xu, *Member, IEEE*, and Xiaotong Zhang, *Senior Member, IEEE*

***Abstract*—With the advent of the era of data-driven material R&D, more and more countries have begun to build material Big Data sharing platforms to support the design and R&D of new materials. In the application process of material Big Data sharing platforms, storage and retrieval are the basis of resource mining and analysis. However, achieving efficient storage and recovery is not accessible due to the multimodality, isomerization, discrete and other characteristics of material data. At the same time, due to the lack of security mechanisms, how to ensure the integrity and reliability of the original data is also a significant problem faced by researchers. Given these issues, this paper proposes a blockchain-based secure storage and efficient retrieval scheme. Introducing the Improved Merkle Tree (MMT) structure into the block, the transaction data on the chain and the original data in the off-chain cloud are mapped through the material data template. Experimental results show that our proposed MMT structure has no significant impact on the block creation efficiency while improving the retrieval efficiency. At the same time, MMT is superior to state-of-the-art retrieval methods in terms of efficiency, especially regarding range retrieval. The method proposed in this paper is more suitable for the application needs of the material Big Data sharing platform, and the retrieval efficiency has also been significantly improved.**

***Index Terms*—Materials genome big-data, multi-source heterogeneous data, data storage, data retrieval, blockchain, merkle tree.**

Manuscript received 16 December 2022; revised 19 April 2023; accepted 6 July 2024. Date of publication 10 July 2024; date of current version 25 July 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3702403, in part by the National Natural Science Foundation of China (NSFC) under Grant 62101029, and in part by China Scholarship Council Award under Grant 202006465043 and Grant 202306460078. Recommended for acceptance by M. Fazio. *(Corresponding authors: Cheng Xu; Xiaotong Zhang.)*

Ran Wang is with the School of Computer and Communication Engineering, Shunde Innovation School, University of Science and Technology Beijing, Beijing 100083, China, and also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: wangran423@foxmail.com).

Cheng Xu is with the School of Computer and Communication Engineering, Shunde Innovation School, University of Science and Technology Beijing, Beijing 100083, China, and also with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: xucheng@ustb.edu.cn).

Xiaotong Zhang is with the School of Computer and Communication Engineering, Shunde Innovation School, University of Science and Technology Beijing, Beijing 100083, China (e-mail: zxt@ies.ustb.edu.cn).

Digital Object Identifier 10.1109/TPDS.2024.3426275

I. Introduction

With the rapid expansion of the market scale of the Big Data industry, the material industry is also taking advantage of Big Data to make revolutionary changes. Massive material data can be analyzed, mined, and utilized through Big Data technology to obtain corresponding values. Materials R&D also means gradually entering the fourth paradigm, data-driven materials R&D [1], which can reduce costs and significantly shorten the R&D cycle. Therefore, more and more countries worldwide have begun to build data-sharing platforms and other infrastructure, provide valuable data for researchers, and support the design, research, and development of new materials.

Due to the multimodality, isomerization, and discreteness of material data [2], various materials have their own composition and performance concerns. Even the same kind of material may exist in different databases in distinct structural forms. The severely fragmented, isomerized, and decentralized material data makes it very difficult to store and retrieve, slowing down the development of resource integration, sharing, and analysis in the material field. At the same time, it is urgent to take necessary security measures to protect sensitive data, as they may involve economic growth and even national security. However, most of the existing material Big Data platforms only provide fundamental security measures, such as identity authentication and access control. Once a leak occurs, it isn't easy to trace back to the source of the event. Protecting the original data from being leaked and tampered with is a critical problem that researchers must solve.

The vigorous development of blockchain technology has provided corresponding solutions for data protection of distributed Big Data platforms [3], [4], [5]. Based on its tamper proof, traceable and auditable characteristics [6], blockchain has been adopted in finance, the Internet of Things, the Internet of Vehicles, and other fields to ensure the security and reliability of distributed system data storage [7]. However, the service provider must completely traverse the entire blockchain to obtain the query results in the retrieval process. At the same time, the blockchain is a distributed ledger that continuously adds transaction records [8], and the height of the blockchain will also increase with the number of transaction requests. As a result, the response cycle of retrieval will become longer and longer and ultimately can not match the retrieval requirements of the application.

To solve the above problems, this paper proposes a safe and efficient data access scheme by introducing the Improved Merkle Tree (MMT) based blockchain to solve the applicability, efficiency, and security problems faced in the process of multi-source heterogeneous material data storage and retrieval. The main contributions of this paper include the following three folds:

1) A blockchain-based Big Data security distributed storage framework is proposed, which can effectively solve the security problems faced by multi-source heterogeneous material data in the storage and retrieval process. In terms of storage, each participant can flexibly deploy blockchain nodes without changing the underlying database framework. Distributed ledgers can ensure data storage security and achieve tamper-proof, traceability, and auditability. In terms of retrieval, the integrity of retrieval results is verified by the hash value stored on the blockchain. At the same time, the proposed method only requires one data tamper-proof verification, which greatly improves the efficiency and reduces the computing overhead.
2) Given the heterogeneous characteristics of material data, a dynamic container model is proposed to solve raw data's off-chain storage. Users can customize templates applicable to their material data structures. The dynamic container model normalizes the collected original data sets and converts them into container data sets through templates, which are resolved into different data structures by database adapters and stored in appropriate databases. The standardized data set will be helpful for subsequent material data retrieval and calculation analysis. Furthermore, the final results are displayed to users as template structure reorganization.
3) By adding an improved Merkle tree structure to the blockchain, the transaction data on the chain and the original data stored in the off-chain cloud are mapped through templates to achieve efficient material data retrieval. MMT combines the advantages of Merkle Tree and Merkle Patricia Trie (MPT), does not need to make any changes to the underlying storage structure, and is suitable for material data retrieval based on template structure. In addition, compared with state-of-the-art, our proposed method has higher efficiency.

The rest of this paper is organized as follows. Section II introduces the material Big Data platform and the storage and retrieval scheme based on the blockchain framework. Section III proposes a blockchain-based storage scheme for multi-source heterogeneous material data from both the off-chain and on-chain dimensions. In Section IV, an efficient data retrieval scheme based on MMT and a verification method for the integrity of raw material data are proposed. Section V analyzes the impact of MMT structure on blockchain performance and compares it with state-of-the-art. Section VI summarizes the full text and prospects.

II. Related Work

A. Material Big-Data Platform

Currently, many material Big Data platforms have been used at home and abroad. From a data management and storage perspective, most of these platforms adopt centralized storage. The material database of the National Institute of Standards and Technology [9] can accept data in any format, providing a feasible scheme for exchanging and reusing material data. The storage mode of the material database is relatively simple and direct, and the "as is" data provided by the data provider is used for storage. Therefore, a variety of data are stored in the material database. Still, due to the extreme heterogeneity of the stored data, the data content cannot be directly retrieved or integrated with the analysis tools. Only "as is" data can be provided to the demander.

AFLOW [10], [11], [12], Open Materials Database [13], and MARVEL NCCR [14] store material data in different types of databases, such as relational databases, file systems or MySQL. AFLOW only supports retrieval by the Inorganic Crystal Structure Database Number (ICSD Number for short), Aflowlib Unique Identifier (AUID for short), and string but does not provide range retrieval, fuzzy retrieval, and other functions. The Open Materials Database provides a more straightforward retrieval function, which can only be retrieved by keywords. The retrieval results are displayed according to the classification of "optimization results" and "contribution results," and the number of display results is limited to 100. MARVEL NCCR does not provide a retrieval function for users to enter keywords but only provides an index list of data sets in the database, including relevant articles, videos, codes, and other information published in the material field. Therefore, it isn't easy to quickly locate the information content related to a keyword.

On the other hand, MDF [15], Materials Project [16], and OQMD [17] adopt a distributed storage scheme to store big material data in local databases or cloud servers of different institutions, which improves storage performance. However, the data storage structures of different institutions are not uniform, which increases the difficulty for the subsequent sharing of multi-source heterogeneous material data. MDF provides keyword and range retrieval according to resource type, label, organization, release year, and other types. The retrieval results are some summary information, and users must authenticate to obtain the original data. The Materials Project requires users to perform identity authentication before retrieving data. Keywords to be retrieved should be input according to a given format, including only certain types of elements, at least certain types of elements, and formulas. The OQMD allows users to retrieve data according to the fields defined in the template according to user-defined template, generate a retrieval file in a fixed format, and query relevant data by uploading the retrieval file. The advantage is that it can obtain more accurate data, while the disadvantage is it requires a good understanding of the data structure of the platform, increases the user's learning cost for using the platform, and lacks flexibility.

It can be seen that access to material data is still the main problem to be solved by the material Big Data platform. Researchers also focus on ensuring the safety of material data in storage, retrieval, analysis, and sharing. It is necessary to consider how to apply the characteristics of multi-source heterogeneous data of material Big Data to achieve efficient and secure storage and retrieval.

B. Blockchain-Based Data Storage

Blockchain provides a new solution for the secure storage of multi-source heterogeneous material data [18], which can promote the centralized audit and management of the underlying database. Blockchain was originally designed for digital currency, represented by Bitcoin [19]. It created a complex consensus mechanism, such as proof of work, which requires substantial computing resources, making the blockchain system unsuitable for large-scale data storage applications. To solve these problems, Muzammal et al. [20] combined the blockchain with the database to build a log-based application platform, which can provide multi-active database and multi-disaster recovery middlewares. Yue et al. [21] used the blockchain to provide data storage solutions for parties with data sources, where data is linked to blocks in a specific way. Liu et al. [22] proposed an efficient data collection and sharing scheme based on blockchain and created a reliable storage environment in combination with Ethereum and Deep Reinforcement Learning (DRL).

However, storing all raw data on the blockchain is not optimal. The consensus mechanism of the blockchain takes up a lot of resources. The platform's overall performance will be affected if all material data is uploaded to the chain. Therefore, "storing transactions on the chain and raw data off the chain" will be an alternative to ensure data storage security and performance without affecting throughput and computing performance.

C. Blockchain-Based Data Retrieval

Some retrieval algorithms have been proposed to improve data retrieval ability in the blockchain and optimize them from different aspects to improve retrieval efficiency.

Some research improves retrieval efficiency by optimizing the data structure. VChain [23] proposed an accumulator-based authentication data structure, which supports verifiable Boolean range queries on the blockchain. At the same time, an inverted prefix tree structure is proposed to speed up the processing of massive subscription queries. Xu et al. [24] put forward a CMPT structure based on MPT by introducing Lock free (Compare And Wap, CAS) and Cache arrays, to realize parallel operations. Multiple pieces of data can be inserted simultaneously, and the retrieval efficiency can be improved. Qu et al. [25] proposed a retrieval method based on temporal and spatial dimensions to improve retrieval efficiency by adding appropriate fields in the block header. Generally, these techniques only apply to some specific scenarios or retrieval functions.

Some aims to improve retrieval efficiency by changing the block structure. Jia et al. [26] took the AB-M tree as the storage structure of the transaction and added Min and Max fields to the block header, combining the advantages of a balanced binary tree (fast data retrieval) and Merkle tree (fast data verification) to provide speedier transaction retrieval and ensure the integrity of the transaction. However, its application is limited to online transaction retrieval, lacks association with data outside the chain and does not propose appropriate optimization methods for range retrieval. Pei et al. [27] created a mapping between on-chain and off-chain data by adding a Merkle semantic tree structure to the block, thus realizing real-time data retrieval. At the same time, multiple complex analysis query primitives are provided to support semantic, queries, and even fuzzy queries. Although the above algorithm optimizes block structure and does not need to traverse the entire blockchain to retrieve data, the high computing overhead remains a problem.

Another part of the research improves retrieval efficiency by modifying the blockchain framework. Muzammal et al. [20] proposed ChainSQL by deploying the database for each node in the blockchain and then verify the results from the blockchain. Wu et al. [28] proposed a verifiable query layer (VQL) deployed in the cloud to provide efficient and verifiable data query services for the blockchain system. To prevent forged data, the database maintains an encrypted fingerprint and writes it to the blockchain to ensure the authenticity. Li et al. [29] developed EtherQL to provide retrieval capability by adding an Ethereum query layer. EtherQL provides efficient query primitives for analyzing blockchain data, including range query and top-k query. Although the above methods can improve retrieval efficiency by querying directly from the database, the computational cost of verifying the authenticity is significant, and they are not suitable for with large scale applications.

III. Secure Data Storage

This paper uses the blockchain to store all transaction data, while the platform's Cloud Servers store the original material data. At the same time, due to the characteristics of multi-source and heterogeneous material data, a template-based data collection method is proposed. Data with different structures in the template are stored in various databases in the background through dynamic containers. On-chain data is also classified and stored in template type. It can be seen that the data stored on the chain and off the chain depend on the material data template as a bridge to connect. The organic combination of on-chain and off-chain storage is achieved through the design of data storage transactions.

A. System Storage Framework

From the perspective of the blockchain-based storage framework, it could be divided into the following components, cloud server, light nodes, and all nodes, as shown in Fig. 1. The cloud server stores raw material data, consisting of different databases, such as PostgreSQL, MongoDB, MongoDB's GridFS, etc. All transaction records are stored on the blockchain, and the light node only stores all block header information on the blockchain. The full node stores the complete blockchain ledger, including all block headers and bodies, which can verify the integrity of the original data retrieved, and is responsible for the broadcast and verification of blockchain transactions. To improve the retrieval efficiency on the chain, the keywords of the original data in the

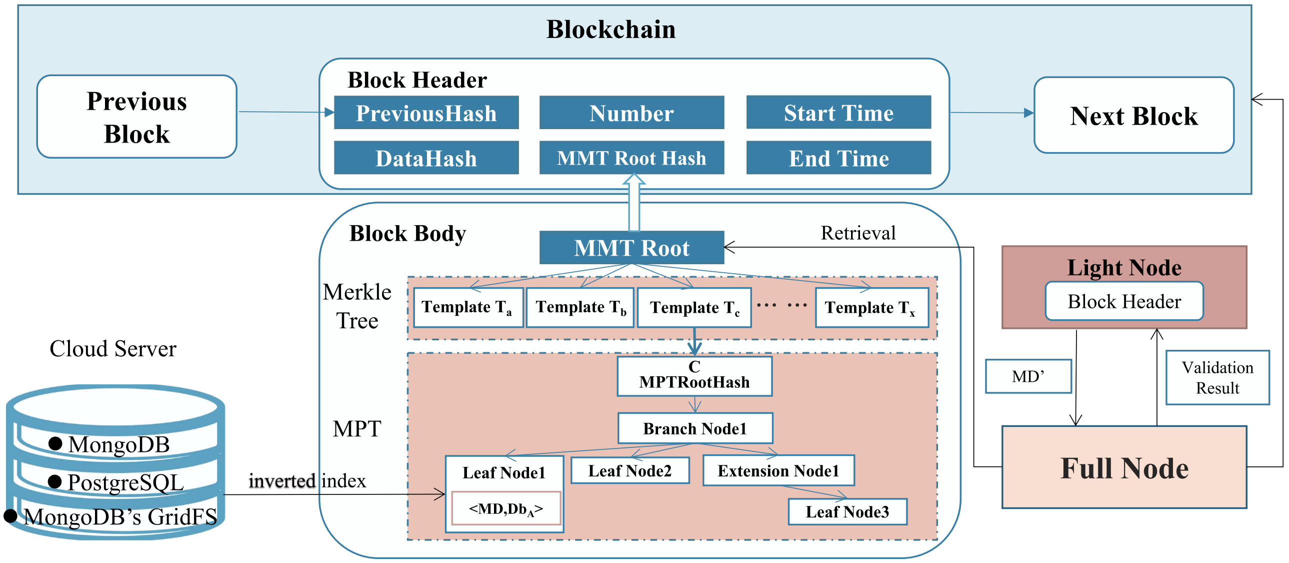


Fig. 1. A framework diagram of the overall storage system.

data storage system are extracted through ElasticSearch [30] in the off-chain data storage system, and the inverted index table is built and transferred to the blockchain. The index structure on the chain is rebuilt in MMT structure according to the inverted index table to facilitate subsequent on-chain retrieval.

From the perspective of a single block, block $B = (B_{header}, B_{body})$ consists of block header B_{header} and block body B_{body}, where B_{body} stores the MMT structure, and the leaf node stores data in the form of a key-value pair. The value includes not only the metadata of the transaction but also the storage address of the original data (Database Address, recorded as DbA), which facilitates the subsequent search. To improve the retrieval efficiency and provide the function of range retrieval based on time and template type, we added new fields to the blockchain's original B_{header} structure, mainly including the following two parts:

1) Two additional fields, $StartTime$ and $EndTime$, have been added. Temporal range-based retrieval is one of the most basic retrieval conditions. The original blockchain provides a key-history index, but the retrieval efficiency is relative low, while the time complexity is $O(n^2)$. By adding a time field in the block header, the time to obtain the transaction timestamp is saved, and the retrieval efficiency can be improved. In addition, the binary search's complexity is $O(\log(n))$, which greatly improves the retrieval efficiency, helps quickly locate the data in a certain period, and speed up the process of material data analysis.
2) $MMTRootHash$ has been added. The time complexity of keyword retrieval is reduced from $O(n^2)$ to $O(\log(n))$, which saves the time of deserializing transaction records to extract metadata. On the other hand, it provides a range search function based on the material data template. Due to the heterogeneous characteristics, material data is collected based on templates, and data with different structures are stored in various databases via dynamic containers (see Section III-B for details). Therefore, the construction of an MMT structure based on the template type has strong applicability to both the off-chain storage system and the material data retrieval on the chain.

B. Off-Chain Storage: Dynamic Container

Material datasets are usually heterogeneous and stored in different custom formats. Data sets often need to be standardized via universal templates, finally achieving accurate retrieval and computational analysis to reduce users' cognitive burden and learning costs. Therefore, this paper will describe the off-chain storage scheme of multi-source heterogeneous material data from material data classification, the composition of dynamic containers, and the creation of containerized data sets.

The function and practicability of the storage scheme primarily count on the number and complexity of material data types. Based on the characteristics of material data, material data could be divided into *basic types* and *composite types*. The basic types include string, numerical, picture, and file types. A composite type comprises various basic types: range, selection, array, table, generator, container type, etc. Range type consists of a numerical type, representing the interval value of two numerical values; The selection type is composed of string type, depicting a text option that can be selected for an attribute; The array type is composed of an arbitrary basic type, representing an ordered list of values; Table type, generator type, and container type are composed of different sets of basic types. The difference between these three is the form of type value selected in the sets. The generator type can only take one value of certain types in the sets. Container type can take a value of all types in the sets. Table type can take any number of values of all types in the sets.

Based on the type of material data, a Dynamic Container Model (DCM) is designed to adapt to the storage of multi-source heterogeneous material data. The built-in structure of abstract containers in DCM is designed to be dynamically constructed

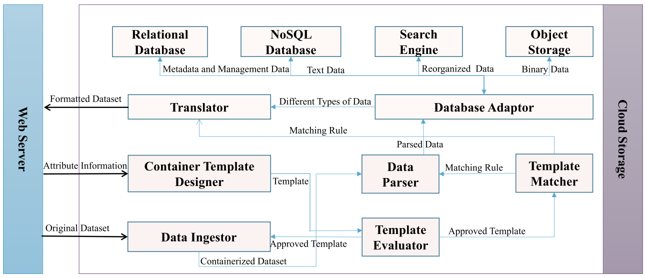


Fig. 2. Schematic diagram of relationship among components in the dynamic container.

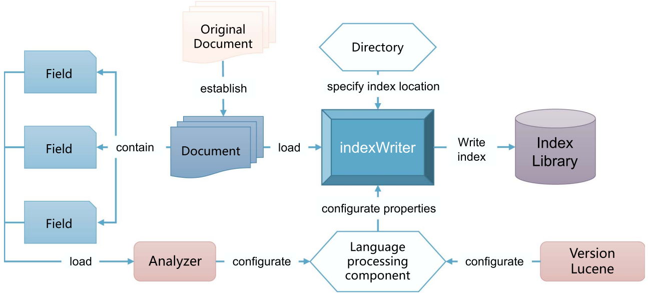


Fig. 3. A flowchart of the inverted index creation.

from different types of material data. Therefore, DCM provides a method for storing, packaging, and exchanging data and allows users to customize templates suitable for data structures.

From the composition of dynamic containers, DCM mainly includes the following components: data ingestor, container template designer, template evaluator, template matcher, and data parser. The user can customize the template through the container template designer, and the template evaluator will audit the template. The data ingestor uses the approved template to normalize the original dataset and convert it into a containerized dataset. The data parser and template matcher will parse the containerized dataset into metadata, text material data, binary files, and other formats, which will be stored in the appropriate database by the database adapter. The standardized dataset will be helpful for subsequent material data retrieval and calculation analysis. When the underlying database needs to provide the web server with the retrieved raw data, the translator matches the template rules, reassembles the material data of different structures, and presents the formatted data in the template type. The relationship diagram of the dynamic container is shown in Fig. 2.

From the perspective of creating a containerized dataset, the containerized dataset consists of two main parts: the container template and the container instance. The container template represents an abstract description of the properties and structure of a material dataset. We denote ":" to describe the relationship between the property and the data type. A type declaration expression is marked as "$x : T$", which means the property x is of type T. A container template S contains a series of expressions of data-type declaration, denoted as:

$$S = \left\{ x_i : T_i^{i=\{1,\ldots,n\}} \right\} = \{x_1 : T_1, \ldots, x_n : T_n\} \tag{1}$$

where x_i indicates the properties and T_i indicates the name of date-types. A container instance represents an abstract description of the data gathered together. It specifies the value of each property and is constrained by the dataset template. The assignment expression "$x = v$" indicates that the property value x is v at some point. Then, the container instance C could be represented by a series of assignment expressions:

$$C = \left\{ x_i = v_i^{i=\{1,\ldots,n\}} \right\} = \{x_1 = v_1, \ldots, x_n = v_n\} \tag{2}$$

Then, a normalized description of a material dataset could be described as a containerized set (S, D), comprising a template and several instances, where $D = \{C_i^{i=\{1,\ldots,n\}}\} = \{C_1, \ldots, C_n\}$. It can be seen that DCM uses the template to normalize and convert the original data set into a containerized one. The standardized dataset will facilitate subsequent retrieval and computational analysis of material data.

C. *On-Chain Storage: MMT*

To facilitate efficient retrieval based on blockchain in the future, this paper creates an MMT structure through the inverted index, index pointers between blocks, Merkle tree, and MPT to store $< key, (MD, DbA) >$ key-value pairs according to the template type. This section describes the storage scheme on the chain from the aspects of inverted index creation, MMT structure, and MMT creation.

1) Inverted Index Creation: Unlike the structured database system, a large amount of data stored in the blockchain has no fixed structure, so the commonly used hash index [31] and B+ tree index [32] in the database cannot be directly applied to the blockchain system. Most full-text indexing technologies are based on inverted indexing [33]. It supports keyword query, the most effective index structure for multi-source heterogeneous data. Therefore, in the retrieval scheme, this paper uses the inverted index to obtain the corresponding relationship of $< key, (MD, DbA) >$ in the database system. We take ElasticSearch [30] (referred to as ES) as an example and use different databases in the cloud server to store material gene Big Data with various structures. Then, import the data into the cluster established by ElasticSearch, create a $< key, (MD, DbA) >$ data map, and establish a reverse index to facilitate subsequent retrieval of the original material data. ES is based on the open-source library Lucene [34] and provides application interfaces by encapsulating the relevant functions, a query engine, and a search engine to complete the purpose of data retrieval. Thus, it has good usability. As ES is widely used in the industry, the inverted index construction is not described here but only in the inverted index construction flow chart, as shown in Fig. 3.

According to the original dataset, extract the keywords through ES, build a $< key, (MD, DbA) >$ mapping relationship, and return the key-value pair to the web server for creating MMT. key is the hexadecimal token generated by encryption for each keyword, and $value$ is the keyword's corresponding MD and DbA. Finally, insert the key-value pair $< key, (MD, DbA) >$ into the MMT.

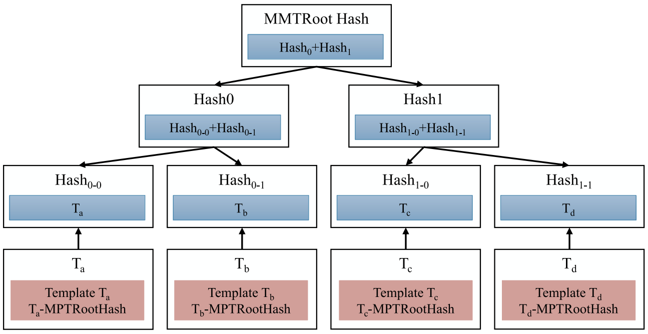


Fig. 4. The workflow diagram of data uploading.

2) MMT Structure: MMT is mainly composed of the Merkle tree and the MPT. The upper layer of MMT uses the Merkle tree to classify data according to the template type to support range queries. At the same time, each leaf node of the Merkle tree stores the root hash of all data under a specific type of template, as shown in Fig. 4. The values stored in the leaf node are recorded as $< Template\,TId, TId - MPTRootHash >$, $TId \in TemplateSet$. $TemplateSet$ is the approved template set of the platform. The root node of the Merkle tree is recorded as MMT Root Hash and stored in the B_{header} for subsequent data authenticity verification. In this paper, data is classified and stored based on templates, and then metadata corresponding to keywords is found from such data templates. This method matches the data collection mode, where material data is submitted according to the template type during the collection process. Therefore, the retrieval algorithm based on MMT is more conducive to the multi-source heterogeneous material data, facilitating researchers to search and analyze based on a specific type of material data template and speeding up the developing process of new materials.

The lower layer of MMT adopts the MPT to support keyword queries, with the functions of searching, verifying, and storing the key-value pair simultaneously. The key is a particular hexadecimal encoding of the keyword, and $value$ is the Recursive Length Prefix (RLP) [35] encoding. There are four types of nodes in an MPT tree: root node, leaf node, branch node, and extension node. The root node stores the hash of the whole tree, which is consistent with the data structure of the extension node. The leaf node is responsible for storing key-value pairs. The extension node is also responsible for storing key-value pairs, but its key is the common prefix of its child nodes, and the value is the hash of all its child nodes. Finally, the branch node has 17 elements. The first 16 elements represent the node's possible branches (16 hexadecimal digits), which are also the first unequal hexadecimal digits in the key. The 17th element is the value field, the hash of all child nodes. All kinds of nodes are connected through pointers. From the $sharednibble$ of the root node to the $keyend$ of the final leaf node, a complete key is formed. The $value$ corresponding to the key is stored in the leaf node. The structure of the MPT is shown in Fig. 5.

3) MMT Creation: From the view of creating the upper Merkle tree, two possible scenarios exist for inserting key-value pairs into the MMT. First, if the data template of the keyword

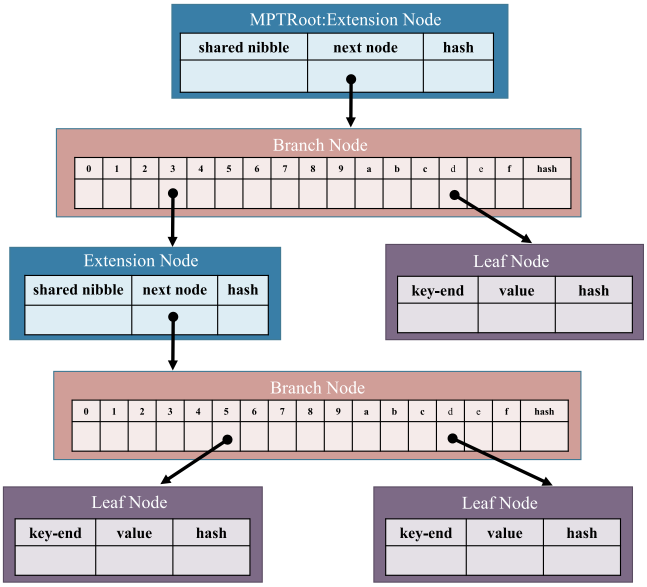
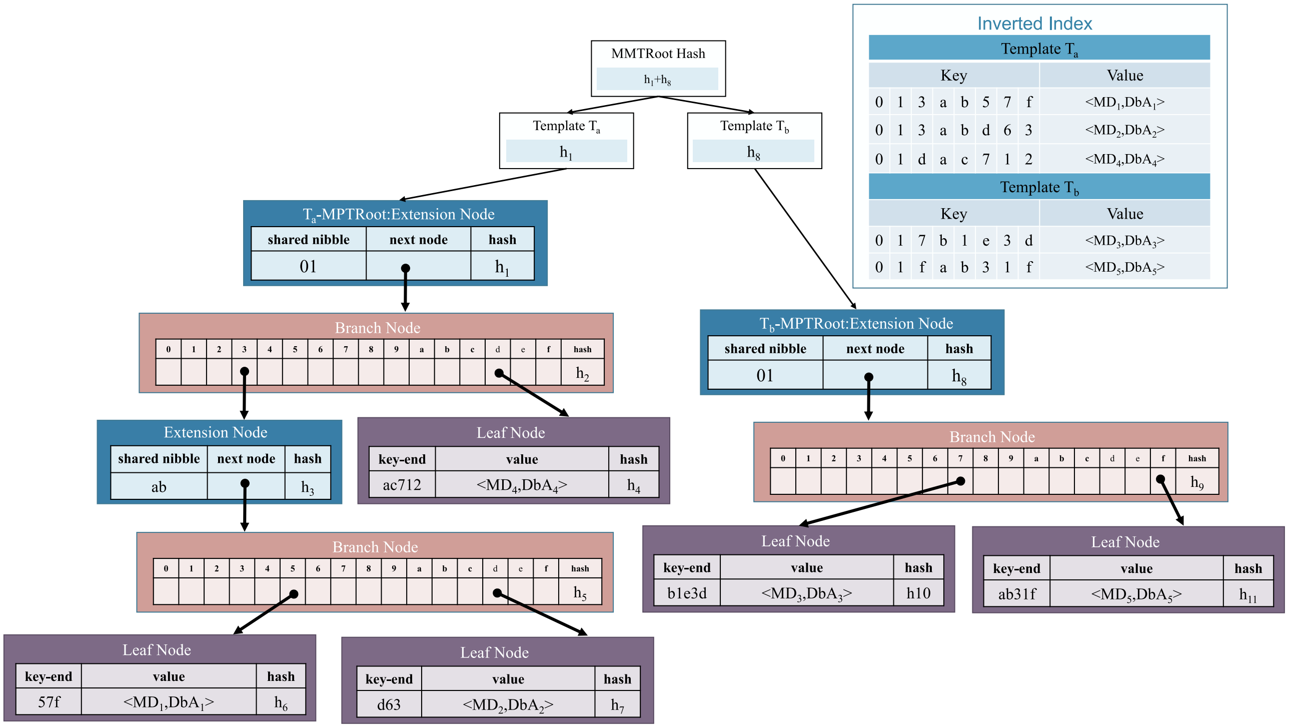


Fig. 5. The MPT structure in MMT.

to be inserted does not match that in the Merkle tree's leaf node on the upper layer of the MMT, then a new Merkle tree leaf node should be created. For example, the creation process of the Template T_b branch is displayed in Fig. 6. When the keyword $key_3 = 017b1e3d$ is inserted, $MD_3.\,Template = T_b$ does not match any leaf node when traversing the upper Merkle tree of the MMT. Therefore, a new leaf node needs to be added to the Merkle tree, that is, the root of the MPT belonging to Template b. The newly created MPT stores all key-value pairs belonging to Template T_b. The second case is when the data template of the keyword to be inserted matches the corresponding template in the leaf node of the Merkle tree, the matching $T_x - MPT$ will be traversed, where $T_x \in TemplateSet$.

On creating the lower level MPT, the material data template is determined with corresponding to the keyword based on the Merkle tree. Then, start to create the lower level MPT structure of the MMT, and insert the $< key, (MD, DbA) >$ key-value pairs, as shown in Fig. 6. Inserting a key-value pair into a leaf node of MPT also involves two cases: the first is the key to be inserted exactly matches that of the leaf node. In this case, the value of the original leaf node is directly modified, and the new value is appended to the original. For example, the original is $data_1$, and the new is $data_2$. Insert $< key, value >=< key_1, (data_1, data_2) >$ to the leaf node. In the second case, if the key to be inserted does not match that of the leaf node, the matching public prefix should be utilized to generate an extension node. Then, the branch node's multi-branch feature is taken advantage of to point to multiple leaf nodes. Finally, new pairs $< key_2, data_3 >$ and $< key_1, (data_1, data_2) >$ are stored to the original leaf node. The specific algorithm flow of MMT tree creation is shown in Algorithm 1.

Since MMT generally contains all index information on the blockchain, the size of the index increases as the index information increases. MMT introduces a node index pointer structure to reduce the storage cost within the block. If the MMT is not



Inverted Index

Template T_a	
Key	Value
0 1 3 a b 5 7 f	$<MD_1,DbA_1>$
0 1 3 a b d 6 3	$<MD_2,DbA_2>$
0 1 d a c 7 1 2	$<MD_4,DbA_4>$
Template T_b	
Key	Value
0 1 7 b 1 e 3 d	$<MD_3,DbA_3>$
0 1 f a b 3 1 f	$<MD_5,DbA_5>$

Fig. 6. Create MMT based on inverted index table.

TABLE I
FIELDS AND MEANINGS OF DATA STORAGE TRANSACTIONS

Field Name	Type	Definition
isSuccess	boolean	Verify whether the web server signature is successful
Sign	string	Web server sign
MD	MetaData	Data metadata information, see Table 2
Content Hash	string	The hash of data content

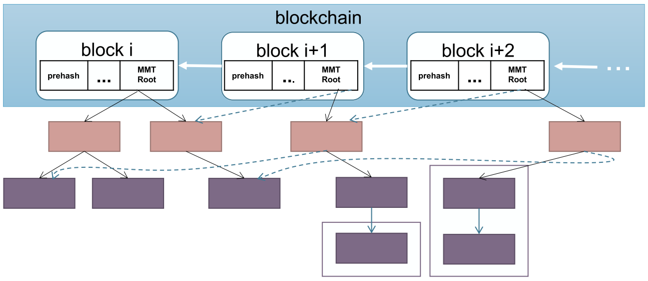


Fig. 7. A schematic diagram of MMT update.

updated, it is not necessary to keep it in the newly generated block. On the contrary, add a node index pointer to the last node in the previous block where the MMT has not changed. Only the updated node needs to be stored in the new block's MMT. Fig. 7 details the process of index updating. The pink node represents the upper Merkle tree of the MMT, the purple node represents the lower MPT, and the purple node in the box is the newly inserted node in the MMT. This update method is also helpful for template-based retrieval, and only the MPT of a specific template type in the block body that meets certain requirements needs to be traversed.

The transaction is a bridge for the interaction of each participating entity of the material Big Data platform. It can establish a mapping relationship between the off-chain data and the on-chain data and build a storage mode of "storing transactions on-chain and original data off-chain."

D. The Transaction Process of Data Storage

The transaction is a bridge for the interaction of each participating entity of the material Big Data platform. It can establish a mapping relationship and build a storage mode of "storing transactions on-chain and original data off-chain."

Data storage transactions mainly include these following fields, namely $isSuccess$, $Sign$, MD, and $ContentHash$, whose specific meanings are shown in Table I. The metadata MD includes 16 fields, such as data ID, data title, data abstract, and template to which the data belongs. The definitions of other fields are shown in Table II. Among them, $Template$ is closely related to the MMT-based retrieval method. Table III shows all the fields and meanings of the material data template.

Five main entities are involved in the entire data storage transaction process: the client, web server, blockchain, LevelDB, and cloud server. The user creates a material data template on the client side that fills in the data form, including metadata, data content, etc. The web server plays the role of middleware. As the forwarder of the requested transaction, it isolates the blockchain

Algorithm 1: MMT Establishment Algorithm.

```
Input: inverted index data set
    $InvSet =< key_n, (MD_n, DbA_n) >$, $n$ as the length of
    inverted index set. $R$ as the MMTRoot node.
for $i = 1$ to $n$ do
    if $InvSet.MD_i.template$ match $R.child_j.template$ then
        flag set up false
        foreach $child_m$ in $< R.child_j >$ do
        if $key_i$ match $R.child_m.key$ then
            insert $< key_i, (MD_i, DbA_i) >$ to $R.child_m$
            flag set up true
        end if
        if flag==false then
            Add value of new node into $R.child_j$
        end if
        end for
    else {$InvSet.MD_i.template$ not match $R.child_j.template$}
        create a new node as $MPTRoot$ insert to $R$, record as $newMPTR$
        insert $< key_i, (MD_i, DbA_i) >$ to $newMPTR.child$
    end if
end for
Update $R$ hash
Output: MMT $MMT$
```

TABLE II
FIELDS AND MEANINGS OF METADATA

Field Name	Type	Definition
Id	int	Data Id, unique
Titlte	string	Data title, unique
Source	string	Source information
Reference	string	Reference information
Methods	string	Data method information
TId	int	Template Id
key	string	Data keyword information
Doi	string	Data Doi information
Abstract	string	Data abstract
Project	string	Project name
Subject	string	Subject name
Contributor	string	User name of data creator
Institution	string	Organization of the data creator
Category	string	Classification of materials to which the data belongs
AddTime	string	Creation time
Template	Template	Template content of data, see Table 3

from the cloud server. All interactions between the blockchain and the cloud server pass through the web server, ensuring the underlying database's security. The blockchain is responsible for authentication, saving the metadata corresponding to keywords and the original database address, and creating data storage transactions to facilitate subsequent data traceability. LevelDB is used for the underlying storage of the blockchain to store and update the MMT structure. Cloud servers are responsible for storing all raw data. The blockchain-based data storage transaction process is detailed in Fig. 8.

TABLE III
FIELDS AND MEANINGS OF TEMPLATE

Field Name	Type	Definition
TId	int	Template Id, unique
TTitle	string	Template title, unique
TCategory	string	The material category the template belongs to
TAuthor	string	Template creator user name
TAbstract	string	Template abstract
TContent	string	Convert template content to string

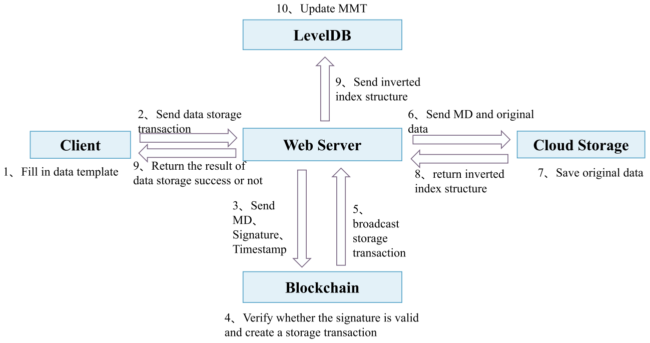

10、Update MMT
LevelDB
9、Send inverted index structure
2、Send data storage transaction
6、Send MD and original data
Client
Web Server
Cloud Storage
1、Fill in data template
9、Return the result of data storage success or not
8、return inverted index structure
7、Save original data
3、Send MD、Signature、Timestamp
5、broadcast storage transaction
Blockchain
4、Verify whether the signature is valid and create a storage transaction

Fig. 8. A flow chart of data storage transaction.

IV. Efficient Data Retrieval Based on MMT

Given that the retrieval efficiency of the blockchain will continue to decline with the increase of the data amount, MMT directly builds the mapping relationship of $< key, (MD, DbA) >$ and no longer needs to traverse the entire blockchain. In addition, it is unnecessary to deserialize all the transaction data inside the block and then extract the metadata, which to some extent improves retrieval efficiency. At the same time, each block only needs to store updated key-value pairs, and MMTs between different blocks are connected through index pointers, saving storage space. The data are associated, which further improves the retrieval efficiency.

A. Retrieval Algorithm Based on MMT

To facilitate the understanding of the MMT-based retrieval, we illustrate the entire process with temporal template-based range retrieval, part of which contains the keyword retrieval, as shown in Algorithm 2.

A given time range $[t_1, t_2]$ is compared with the $StartTime$ and $EndTime$ fields in the block header. As shown in Fig. 9, the creation time of all transactions on the blockchain increases in an orderly manner from the first block to the latest. To save time, we use the binary search to locate the block corresponding to the recent time t_2 and stop the binary search if $t_p \leq t_2 \leq t_q$ is found. Then, traverse forward from the searched block until the block header of $t_m \leq t_1 \leq t_n$ is found, and add all traversed blocks to the retrieval data set $Rset$.

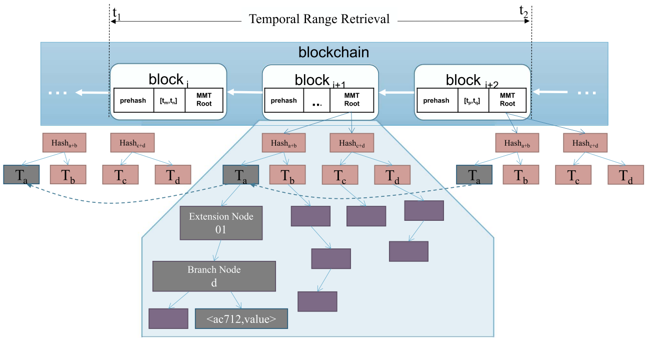


Fig. 9. Schematic diagram of keyword retrieval based on time and template.

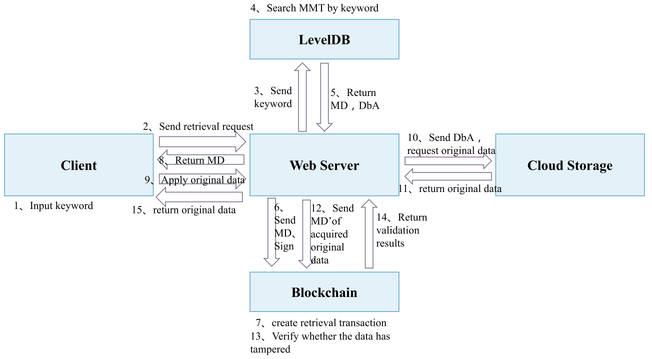


Fig. 10. Data retrieval flow chart.

```
Algorithm 2: MMT Retrieval Algorithm.
Input: Block header $Bh$; MMT $MMT$; retrieval
  conditions:$< Template\,T_a, [t_1, t_2], key_1 >$.
if the node that request the retrieval is a light node then
    communicate with a reliable full node for retrieval
end if
foreach $Bh_i$ in $< Bh >$ do
    if $t_1 \geq Bh_i.StartTime$ and $t_2 \leq Bh_i.EndTime$ then
        record blocknumber and MMTRoot of $Bh_i$ as $Rset$
    end if
end for
foreach $Rset_i.child$ in $< Rset >$ do
    if $Rset_i.child.template = T_a$ then
        Traverse $< Rset_i.child >$
        if $key_1$ is founed then
            retrieval result is written to the set $Mr$
        end if
    end if
end for
Output: retrieval results (MDSet $Mr$)
```

Keyword search: Suppose $Rset =< Block_i, Block_{i+1}, Block_{i+2} >$, taking $Block_{i+1}$ as an example, find the metadata corresponding to the keyword $key_1 = 01dac712$ belonging to the Template T_a in the block body. The block body only stores the key-value pairs of the newly submitted data tothe current block, and connects the nodes of the MMT in all block bodies by pointers to form a complete MMT search tree, as shown in Fig. 9. Therefore, it is more convenient and faster to find the keyword key_1 under Template a. You only need to find Template a first based on the Merkle tree from the latest block, and then find key_1 from the MPT under the template through prefix matching. Template T_a of the parent block could be directly found using pointers, which saves the process of retrieving the Merkle tree. Then, find the keyword position corresponding to the parent node directly, and the retrieval efficiency can be further improved through the pointer structure.

Due to the blockchain's complex structure and network communication, retrieving data through the blockchain is generally slower than that directly from the database. However, our proposed MMT-based method saves the time of finding the hash of block address, traversing the transaction list in the block, and deserializing the transaction to extract metadata. It can query according to the template and the efficiency is close to that of direct retrieval from the database. At the same time, the upper layer of the MMT is based on the Merkle tree, and all nodes store hash values. Thus, the MMT is a tamper-proof structure. The hash value is calculated from the bottom leaf node. Whether any data in the upper or lower MPT is tampered with, the MMTRoot hash value calculated through Hash verification will be inconsistent with the original stored hash. Therefore, the MMT can quickly detect malicious tampering, and the anti-tampering verification is detailed in Section IV-C.

B. Data Retrieval Transaction Process

Retrieval transactions is mainly used to record retrieval results. The structure of retrieval transaction is similar to that of data storage transaction, mainly including the keys, $T_x - rid$, $Template\,TId$, $MDSet$, and $CategorySet$, where $T_x - rid$ represents the unique retrieval record ID, $TemplateID$ represents the unique template ID, $MDSet$ represents the set of metadata corresponding to the retrieval results, and $CategorySet$ represents the material classifications to which the template belongs.

The data demander can enter the time range, template type, and keywords on the retrieval page to initiate a data retrieval process, which also involves five entities, i.e., the client, web server, blockchain, LevelDB, and cloud server. The client is used as the retrieval entrance. The user can input specific content on the retrieval page to initiate the retrieval process, and the retrieval results are also displayed on corresponding page. The web server provides the retrieval application programmable interface (API), which acts as a middleware to forward requests between the blockchain and the cloud server. Blockchain is used to verify the correctness and store retrieval transactions. LevelDB is used to retrieve the corresponding keywords in the MMT and return the related metadata. The cloud server stores the original data and return specific raw data according to the DbA. The data retrieval process based on blockchain is shown in Fig. 10.

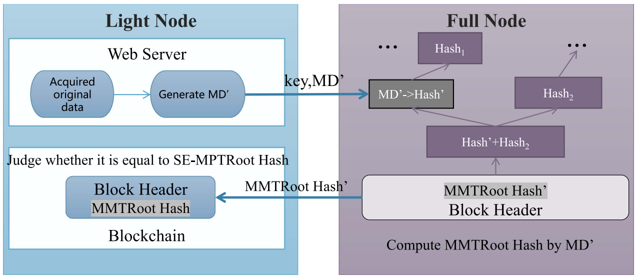


Fig. 11. A schematic diagram of retrieval result verification process.

TABLE IV
EXPERIMENTAL CONFIGURATIONS

Configuration	Parameters
vCPU	11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz
Operating System	Ubuntu 20.10
Number of Virtual Machines	5
Memory	16G DDR3 RAM
Language	Go 1.18
Testing Tool	Apache Jmeter

C. Tamper Proof Verification

The metadata obtained by the data demander can be stored in the blockchain's LevelDB to ensure that it will not tamper with. In addition, ensuring that the original data obtained from the cloud does not tamper with is the primary concern of this article.

If the Full node initializes the retrieve, the authenticity of the original data can only be verified within itself. However, if a Light node only stores B_{header}, it cannot be used for tamper-proof verification inside the node. In this case, it is necessary to communicate with all nearby reachable nodes, while the reachable Full node help verifies the results' authenticity. Light nodes generate MD' from the original retrieved data through the web server and send keywords and MD' to the Full node. Locate the node location corresponding to the MMT via the keyword, and then calculate all hash values on the path from this location to the root, finally obtaining $MMTRootHash'$. Return $MMTRootHash'$ to the Light node to determine whether it equals the $MMTRootHash$ stored in it. If equal, the original data obtained could be proved to have not been tampered with. The schematic diagram of the retrieval result validation process is shown in Fig. 11.

Compared with the blockchain's verification, our proposed method significantly reduces the number of verifications. In the original blockchain structure, the integrity of the search result must be verified by calculating the root hash of the Merkle tree stored in the full node. Since the metadata stored in the MMT only contains some summary information of the original data, there is no need to verify it. Only one tamper-proof verification is required when obtaining the original data.

V. Experiment and Analysis

In this section, we first test the performance of the storage and retrieval framework proposed in this paper, mainly considering the efficiency of blockchain generation and the response time and throughput of the MMT in retrieval. Second, we compared with improved retrieval algorithms based on the Merkle tree, such as CMPT [24], MST [27], AB-M [26], to reflect the advantages of our proposed method.

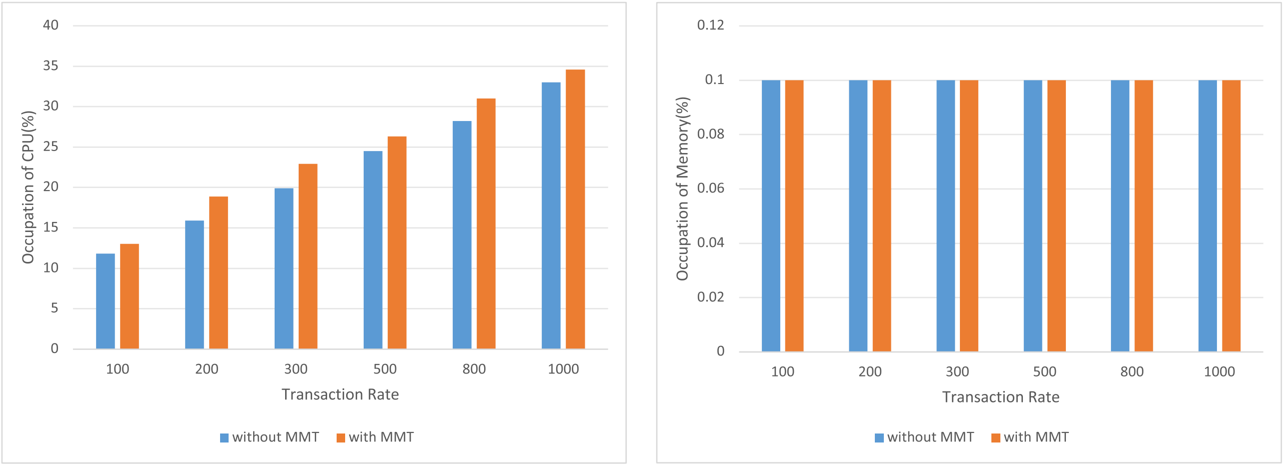
A. Environment Configuration

The material genome Big Data sharing platform we built [36] has been in stable operation for six years, in which over 14 million pieces of material scientific data are stored, and over 2300 data templates are available. The experiment in this paper is carried out in its testing system. The material data type is steel materials, including 10,000 pieces of material data and 20 material data templates. Specific blockchain configuration items and parameters are shown in Table IV. In terms of testing tools, we use Jmeter to test the blockchain-based retrieval performance and simulate user upload and retrieval operations by sending HTTP requests.

B. Performance Evaluation

Since the secure material genome big-data sharing platform does not have too many restrictions on the storage space, the primary purpose of the MMT is to improve the overall retrieval efficiency, which can be achieved by sacrificing a certain amount of storage space. Therefore, this paper focuses on the impact of MMT on retrieval performance. We assume that the transaction request rate is set to 100, 200, 300, 500, 800, and 1000 tps, respectively, and each block contains ten transactions. We tested the performance of the blockchain framework based on MMT from two aspects, including 1) CPU and memory usage during data retrieval in the application scenario of the material big-data platform [36]; 2) The delay and throughput performance of the blockchain with and without the MMT is measured to check whether its performance falls within an acceptable range.

Fig. 12 compares the CPU and memory occupied by the blockchain framework running on cloud servers before and after adding the MMT structure. Fig. 12(a) shows that as the data retrieval transaction request rate increases, the CPU's concurrent tasks need to handle continue to grow. Regardless of whether the MMT structure is added or not, the CPU usage of the blockchain framework will continue to increase. After adding the MMT structure, the web server needs to read LevelDB to obtain the metadata to reduce computational complexity and exchange space for time, which increases communication overhead. At the same time, verification of the integrity of the retrieval results has been added after the retrieval is completed, and the verification process will also increase the CPU workload. Therefore, adding an MMT structure will slightly increase CPU usage. In practical applications, the transaction request rate of the material big-data platform generally does not exceed 200 tps. At a transaction request rate of 200 tps, the CPU utilization rate is only 18.8%, which is only 2.9% higher than that without the MMT, fully



(a)

(b)

Fig. 12. A comparison of server resource usage before and after adding MMT structure. (a) CPU. (b) Memory.

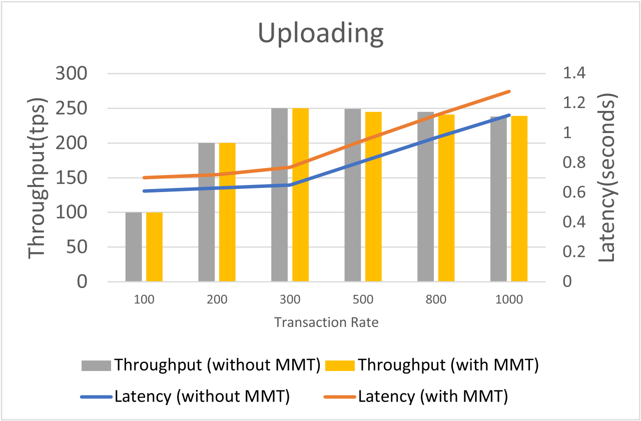


Fig. 13. A comparison of blockchain performance before and after MMT structure is added.

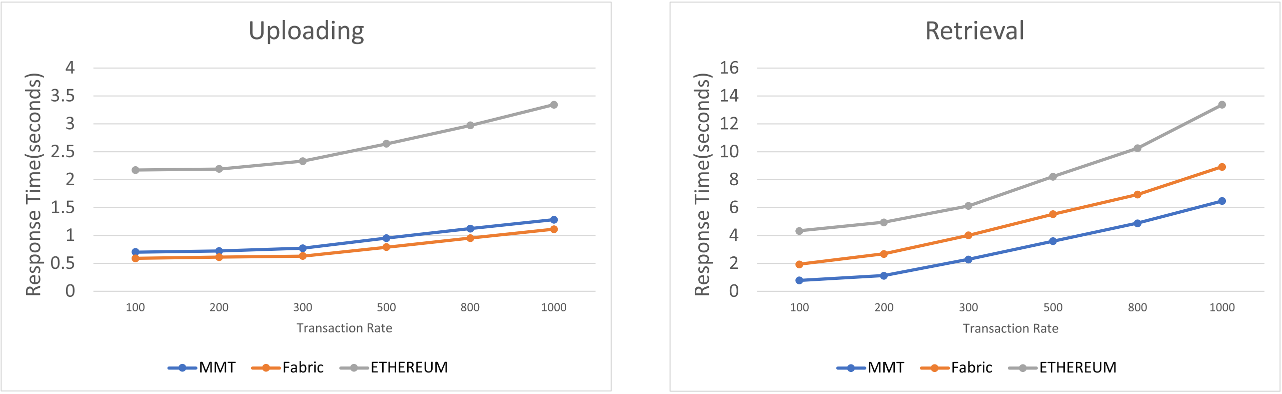
meeting the application requirements of the material Big Data platform. As shown in Fig. 12(b), with the increase of the data retrieval transaction request rate, the memory usage of the blockchain framework before and after adding the MMT structure has remained at 0.1% during operation, and there has been no significant change. This indicates that the blockchain framework does not have high memory requirements during operation, only requiring specific CPU performance.

Fig. 13 compares blockchain performance before and after adding the MMT structure under various data upload transaction request rates. It can be seen from the experimental results that, with the increasing transaction request rate, the changing trend of blockchain performance before and after adding the MMT structure is consistent. When the transaction request rate is lower than 300 tps, there is no significant impact on throughput and latency. When the transaction request rate exceeds 300 tps, the average delay of the blockchain increases with the increase of the transaction request rate. The throughput decreases synchronously, which is stable above 230 tps. It can be seen that the blockchain network can handle transaction requests up to 300 tps without significant network latency. In the practical use of the material Big Data platform, the transaction request rate generally does not exceed 200 tps. At the same time, adding the MMT structure only increased the delay time by 0.09 seconds on average under the transaction request rate of 200 tps. The increased delay time was almost insensitive to the application experience, and MMT also has little impact on the throughput. From this point of view, the MMT structure introduced in this paper has not significantly impacted the blockchain's performance, which is mainly related to the block-out strategy and the hardware carrying capacity of the platform.

C. Performance Comparison of Different Blockchain Frameworks

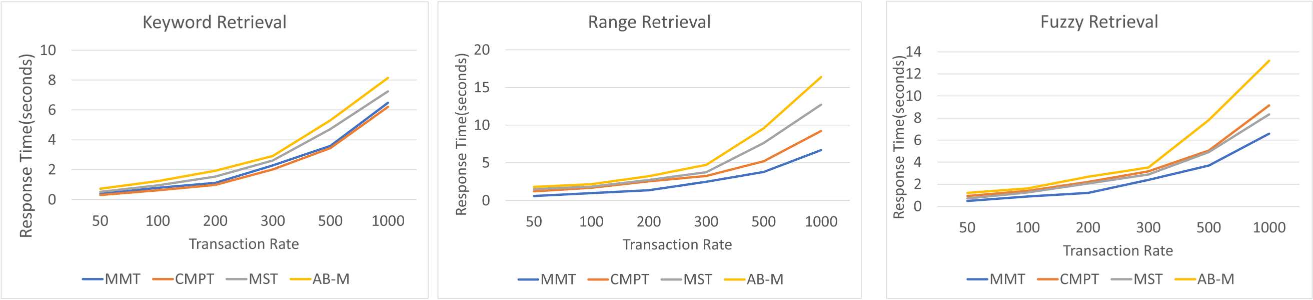
To demonstrate the performance advantages of our proposed MMT-based blockchain framework, we compared it with typical frameworks regarding retrieval efficiency, including Hyperledger FabricTo demonstrate the performance advantages of our proposed MMT-based blockchain framework, we compared it with typical frameworks regarding retrieval efficiency, including Hyperledger Fabric [37] and Ethereum [38].

Fig. 14 compares the performance of MMT-based and other blockchain frameworks in terms of data upload and retrieval transactions. Fig. 14(a) shows that the response times of the three blockchain frameworks remain consistent as the data upload transaction request rate increases. When the transaction request rate is below 300 tps, the response time for data upload increases relatively slowly because the maximum throughput of the blockchain framework has not yet been reached. However, when the transaction request rate exceeds 300 tps, the response time of the blockchain increases with the increase in the transaction request rate. The blockchain framework proposed in this article has a slightly longer response time compared to Fabric



(a) (b)

Fig. 14. A comparison of response times of different blockchain frameworks. (a) Data uploading. (b) Data retrieval.



(a) (b) (c)

Fig. 15. A comparison of retrieval algorithm's response time under various transaction request rates. (a) Keyword retrieval. (b) Range retrieval. (c) Fuzzy retrieval.

due to the addition of an MMT structure, but it is almost imperceptible in application experience. Compared to Ethereum, this framework has a significant advantage in response time because its consensus mechanism has a lower time complexity. Even if an MMT structure is added to the block, the response time for data uploading is still better than Ethereum.

Fig. 14(b) shows that the response times of the three blockchain frameworks for data retrieval transactions continue to grow with the increase in transaction rate. The blockchain framework proposed in this article has significant advantages in retrieval time compared to the other two blockchain frameworks. The average response time of this framework is shortened by 1.82 s compared to Fabric, and by 4.69 s compared to Ethereum. The main reason for this improvement is the addition of an MMT structure, which eliminates the need to traverse the entire blockchain and does not require deserialization operations. It can directly obtain metadata, thus significantly saving retrieval time.

D. Comparison Experiment of Retrieval Algorithms

To verify the performance of the MMT in terms of retrieval efficiency, especially in terms of range retrieval based on material templates, we compared our proposed method with state-of-the-art MPT or Merkle tree-based retrieval algorithms such as CMPT, MST, AB-M, under the same experimental conditions. The requested retrieval transactions rate is set as 50, 100, 200, 300, 500, and 1000 tps, and the response time of various algorithms is compared.

Fig. 15 shows that under different retrieval request rates, the response time of all retrieval algorithms will increase with the increase in transaction rate. As shown in Fig. 15(a), in terms of keyword retrieval, under the condition of 200 tps, the response time of CMPT keyword retrieval is 0.144 seconds faster than that of the MMT method on average. In the process of keyword retrieval, CMPT adds a Cache array structure to increase the time complexity to $O(1)$. MMT does not need to perform the deserialization operation in obtaining metadata, and it can directly get metadata. Thus, the CMPT algorithm does not show significant advantages in keyword retrieval, but the MMT is superior to MST and AB-M algorithms in keyword retrieval efficiency. As shown in Fig. 15(b), in terms of range retrieval, the response time of the MMT method is better than that of others. Under 200 tps, the MMT is 1.168 seconds higher than that of MST. The advantages of MMT range retrieval mainly lie in the following two aspects: on the one hand, it does not need to traverse the block to find the time of keyword creation; on the other hand, the update nodes of the MMT are connected by pointers between blocks, thus reducing the time to find templates and the same keyword in different blocks. As shown in Fig. 15(c), since the

MPT searches keywords by matching the same prefix tree, it is more suitable for fuzzy retrieval. Since CMPT, MST and MMT are all retrieval algorithms modified based on MPT, the efficiency of these three is better than AB-M. In addition, because MMT classifies keywords according to different templates and builds an index structure, it can enable multiple threads to search under different templates simultaneously. Therefore, regardless of the type of retrieval, the MMT has certain advantages over others in terms of efficiency.

The above experimental analysis found that adding an MMT structure has no noticeable effect on the block generation rate. At the same time, MMT is superior to the other three retrieval methods in terms of efficiency, especially regarding range retrieval. The structure of MMT is better suited to the retrieval requirements of the material gene Big Data security sharing platform.

VI. Conclusion and Prospect

This paper proposes a secure storage and efficient retrieval method for genome Big Data of multi-source heterogeneous materials. In terms of security, the introduction of the blockchain framework provides an effective security mechanism for the material Big Data platform in the storage and retrieval process. Regarding storage, multi-source heterogeneous material data is stored in different databases off-chain through material data templates and dynamic container models. In terms of retrieval, the Merkle tree and MPT are combined to build an MMT structure, which can realize keyword, range, and fuzzy retrieval based on templates. At the same time, we further optimized the MMT algorithm through binary search, index pointer, no deserialization, and other methods to improve the overall retrieval efficiency. To verify the integrity of the original data obtained, we confirmed that the original data had not been tampered with through the uniqueness of the hash stored in the MMT. In addition, the experimental results prove that adding an MMT structure has no noticeable effect on the block generation rate. At the same time, MMT is superior to state-of-the-art retrieval methods in terms of efficiency, especially regarding range retrieval. The method proposed in this paper is more suitable for the application needs of the material Big Data sharing platform, and the retrieval efficiency has also been significantly improved.

In the future, the following two aspects should be our primary considerations. On the one hand, since the MMT algorithm can well solve the retrieval problem of multi-source heterogeneous data, we consider expanding it to other application fields to prove its scalability, e.g., edge computing in the industrial Internet of Things [39]. On the other hand, we will further investigate multi-party joint retrieval so that data owners can fully control their material data and not share their original data with data service providers. Instead, multiple material data owners can achieve multi-party joint retrieval through federated learning, multi-party security computing, and other technologies under the condition that the data does not go out of the local area, providing a solid security foundation for data sharing.

References

[1] A. Agrawal and A. Choudhary, “Perspective: Materials informatics and Big Data: Realization of the “fourth paradigm” of science in materials science,” *APL Mater.*, vol. 4, no. 5, 2016, Art. no. 053208.

[2] L. Himanen, A. Geurts, A. S. Foster, and P. Rinke, “Data-driven materials science: Status, challenges, and perspectives,” *Adv. Sci.*, vol. 6, no. 21, 2019, Art. no. 1900808.

[3] M. Macdonald, L. Liu-Thorrold, and R. Julien, “The blockchain: A comparison of platforms and their uses beyond bitcoin,” *Work. Pap*, pp. 1–18, 2017.

[4] H. Yu, H. Sun, D. Wu, and T.-T. Kuo, “Comparison of smart contract blockchains for healthcare applications,” in *Proc. AMIA Annu. Symp.*, 2019, Art. no. 1266.

[5] M. J. M. Chowdhury et al., “A comparative analysis of distributed ledger technology platforms,” *IEEE Access*, vol. 7, pp. 167930–167943, 2019.

[6] A. A. Monrat, O. Schelén, and K. Andersson, “A survey of blockchain from the perspectives of applications, challenges, and opportunities,” *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

[7] N. Deepa et al., “A survey on blockchain for Big Data: Approaches, opportunities, and future directions,” 2021, *arXiv:2009.00858*.

[8] A. Marsalek, T. Zefferer, E. Fasllija, and D. Ziegler, “Tackling data inefficiency: Compressing the bitcoin blockchain,” in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. 13th IEEE Int. Conf. Big Data Sci. Eng.*, Rotorua, New Zealand, 2019, pp. 626–633.

[9] Materials data repository home, 2017. Accessed: Dec. 15, 2022. [Online]. Available: https://materialsdata.nist.gov/

[10] Aflow - automatic flow for materials discovery, 2012. Accessed: Dec. 15, 2022. [Online]. Available: https://aflowlib.org/

[11] S. Curtarolo et al., “AFLOW: An automatic framework for high-throughput materials discovery,” *Comput. Mater. Sci.*, vol. 58, pp. 218–226, 2012, *arXiv:1308.5715*.

[12] S. Curtarolo et al., “ORG: A distributed materials properties repository from high-throughput AB initio calculations,” *Comput. Mater. Sci.*, vol. 58, pp. 227–235, 2012.

[13] Open materials database, 2016. Accessed: Dec. 15, 2022. [Online]. Available: https://openmaterialsdb.se/

[14] S. P. Huber et al., “AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance,” *Sci. Data*, vol. 7, no. 1, 2020, Art. no. 300.

[15] B. Blaiszik, K. Chard, J. Pruyne, R. Ananthakrishnan, S. Tuecke, and I. Foster, “The materials data facility: Data services to advance materials science research,” *JOM*, vol. 68, no. 8, pp. 2045–2052, 2016.

[16] A. Jain et al., “Commentary: The materials project: A materials genome approach to accelerating materials innovation,” *APL Mater.*, vol. 1, no. 1, 2013, Art. no. 011002.

[17] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton, “Materials design and discovery with high-throughput density functional theory: The open quantum materials database (OQMD),” *JOM*, vol. 65, no. 11, pp. 1501–1509, 2013.

[18] Y. Ren, Y. Leng, Y. Cheng, and J. Wang, “Secure data storage based on blockchain and coding in edge computing,” *Math. Biosci. Eng.*, vol. 16, no. 4, 2019, pp. 1874–1892.

[19] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. Accessed: Dec. 15, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[20] M. Muzammal, Q. Qu, B. Nasrulin, and A. Skovsgaard, “A blockchain database application platform,” 2019, *arXiv:1808.05199*.

[21] L. Yue, H. Junqin, Q. Shengzhi, and W. Ruijin, “Big data model of security sharing based on blockchain,” in *Proc. 3rd Int. Conf. Big Data Comput. Commun.*, 2017, pp. 117–121.

[22] C. H. Liu, Q. Lin, and S. Wen, “Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning,” *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3516–3526, 2019.

[23] C. Xu, C. Zhang, and J. Xu, “vChain: Enabling verifiable boolean range queries over blockchain databases,” in *Proc. Int. Conf. Manage. Data*, Amsterdam Netherlands, 2019, pp. 141–158.

[24] J. Xu, Y. Tian, T. Ma, and N. Al-Nabhan, “Intelligent manufacturing security model based on improved blockchain,” *Math. Biosci. Eng.*, vol. 17, no. 5, pp. 5633–5650, 2020.

[25] Q. Qu, I. Nurgaliev, M. Muzammal, C. S. Jensen, and J. Fan, “On spatio-temporal blockchain query processing,” *Future Gener. Comput. Syst.*, vol. 98, pp. 208–218, 2019.

[26] D.-Y. Jia, J.-C. Xin, Z.-Q. Wang, H. Lei, and G.-R. Wang, "SE-Chain: A scalable storage and efficient retrieval model for blockchain," *J. Comput. Sci. Technol.*, vol. 36, no. 3, pp. 693–706, 2021.

[27] Q. Pei, E. Zhou, Y. Xiao, D. Zhang, and D. Zhao, "An efficient query scheme for hybrid storage blockchains based on merkle semantic trie," in *Proc. IEEE Int. Symp. Reliable Distrib. Syst.*, Shanghai, China, 2020, pp. 51–60.

[28] H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "VQL: Efficient and verifiable cloud query services for blockchain systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1393–1406, Jun. 2022.

[29] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "EtherQL: A query layer for blockchain system," in *Database Systems for Advanced Applications*, S. Candan, L. Chen, T. B. Pedersen, L. Chang, and W. Hua (Eds.), vol. 10178. Berlin, Germany: Springer, 2017, pp. 556–567.

[30] A. Yang, S. Zhu, X. Li, J. Yu, M. Wei, and C. Li, "The research of policy Big Data retrieval and analysis based on elastic search," in *Proc. IEEE Int. Conf. Artif. Intell. Big Data*, 2018, pp. 43–46.

[31] G. R. Mitchell and M. E. Houdek, "Hash index table hash generator apparatus," U.S. Patent 4215402, 1980.

[32] X. Li, C. Ren, and M. Yue, "A distributed real-time database index algorithm based on b tree and consistent hashing," *Procedia Eng.*, vol. 24, pp. 171–176, 2011.

[33] G. E. Pibiri and R. Venturini, "Techniques for inverted index compression," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–36, 2021.

[34] Welcome to Apache Lucene, 2000. Accessed: Dec. 15, 2022. [Online]. Available: https://lucene.apache.org/index.html

[35] A. Coglio, "Ethereum's recursive length prefix in ACL2," 2020, *arXiv:2009.13769*.

[36] National meterial data management & service, 2018. Accessed: Dec. 15, 2022. [Online]. Available: http://mged.nmdms.ustb.edu.cn/analytics/

[37] Hyperledger fabric–hyperledger foundation, 2015. Accessed: Dec. 15, 2022. [Online]. Available: https://www.hyperledger.org/use/fabric

[38] Ethereum, 2015. Accessed: Dec. 15, 2022. [Online]. Available: https://ethereum.org

[39] A. Ruggeri, A. Galletta, L. Carnevale, and M. Villari, "An energy efficiency analysis of the blockchain-based extended triple diffie-hellman protocol for IoT," in *Proc. IEEE Symp. Comput. Commun.*, 2022, pp. 1–6.

Ran Wang (Graduate Student Member, IEEE) received the BE degree from the Beijing Information Science and Technology University, China, in 2013, and the MS degree from the University of Science and Technology Beijing (USTB), China, in 2016. She is currently working toward the doctoral degree in the Microarchitecture and Integrated Circuits Lab (MICL) with the University of Science and Technology Beijing. Her research interests include multi-agent systems, distributed security and Internet of Things.

Cheng Xu (Member, IEEE) received the BE, MS, and PhD degree from the University of Science and Technology Beijing (USTB), China, in 2012, 2015, and 2019 respectively. He is currently working as an associate professor in the Microarchitecture and Integrated Circuits Lab (MICL), University of Science and Technology Beijing. He was supported by the post-doctoral Innovative Talent Support Program from Chinese government, in 2019. He is an associate editor of International Journal of Wireless Information Networks. His research interests now include swarm intelligence, multi-robots network, distributed security and Internet of Things.

Xiaotong Zhang (Senior Member, IEEE) received the MS and PhD degrees from the University of Science and Technology Beijing, in 1997, and 2000, respectively. He is a professor in the Department of Computer Science and Technology, University of Science and Technology Beijing. His research includes wireless sensor networks, networks management, signal processing of communication and computer architecture.