2022 19th Annual International Conference on Privacy, Security & Trust (PST) | 978-1-6654-7398-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/PST55820.2022.9851986

Achieving Efficient and Secure Query in Blockchain-based Traceability Systems

Chengzhe Lai, Yinzhen Wang
School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an, China
Email: lcz_xupt@163.com, 18829032031@163.com

***Abstract*—With the rapid development of blockchain technology, it provides a new technical solution for secure storage of data and trusted computing. However, in the actual application of data traceability, blockchain technology has an obvious disadvantage: the large amount of data stored in the blockchain system will lead to a long response time for users to query data. Higher query delay severely restricts the development of block chain technology in the traceability system. In order to solve this problem, we propose an efficient, secure and low storage overhead blockchain query scheme. Specifically, we design an index structure independent of Merkle tree to support efficient intra-block query, and create new fields in the block header to optimize inter-block query. Compared with several existing schemes, our scheme ensures the security of data. Finally, we simulate and evaluate our proposed scheme. The results show that the proposed scheme has better execution efficiency while reducing additional overhead.**

***Index Terms*—Blockchain, Traceability, Efficient query, Secure storage.**

I. Introduction

Blockchain, a practical and revolutionary technology, is widely used in many areas such as data traceability [1]–[6], healthcare, logistics, e-commerce, finance, government administration and recreation. Nowadays, in order to meet practical application scenarios, blockchain-based systems must not only ensure that the stored data is not maliciously modified, but also address the drawbacks in data search and query. The blockchain system has weak data search capabilities. If we want to query a transaction record, the system needs to execute full-text search in the blockchain ledger, which is a very time-consuming process. How to improve the efficiency of data search in blockchain is a crucial issue that needs to be addressed. In addition, the amount of data stored in a block cannot grow indefinitely, and if the block capacity is not limited, it will strictly affect the propagation and storage of blocks.

Several research solutions have been developed to support efficient querying in blockchain systems. Li et al. [7] proposed a system called EtherQL, which can efficiently reorganize the data from the blockchain system to the external database, and realize various query operations through the API provided by the external database. This approach does not modify the original blockchain system. In addition, EtherQL enables automatic and timely synchronization of data blocks and improves data search efficiency by leveraging the power of mangoDB. Peng et al. [8] designed and implemented VQL, a verifiable query middle layer that extracts the data stored in the underlying blockchain system, reorganizes it and provides efficient query services to the upper layer applications. In addition, the middle layer ensures the authenticity of query results. Hiroki et al. [9] proposed a solution based on directed acyclic graphs for the problem of managing historical records in blockchain systems. The solution incorporates the concept of directed acyclic graphs into the design of traceability systems. The solution is able to support merging, branching and splitting while ensuring significant improvements in search efficiency. Xu et al. [10] proposed a lightweight verifiable query framework Vchain, which can support correctness and integrity checking of search query results. In addition, the authors have created two index structures in Vchain, a tree-based index to improve the efficiency of intra-block data search and a skiplists-based index to optimize inter-block data search. Ruan et al. [11]proposed a lineagechain, which creates a new index structure. And the idea is derived from skiplists. The authors combined the lineagechain and forkbase storage systemsin order to improve the efficiency of on-chain provenance queries. Zhang et al. [12] proposed the $GEM^2 - tree$, combining the Merkle tree and B+ tree structures to create the MB tree index. The transaction hash of the original leaf node is replaced with the key, so that the Merkle tree supports keyword search. In Forkbase, Wang et al. [13] proposed a new indexing technique, SIRI, and built a POS tree structure to achieve efficient search of data. This tree structure is similar to a combination of a B+ tree and a Merkle tree. This hybrid structure avoids the cost of consistency maintenance of the original structure under different insertion sequences. Xu et al. [14] proposed a blockchain system that supports fast search, high throughput and low latency. The system builds an MPT-chain model that can support not only historical queries of accounts but also efficient transactional queries.

To address the above problems, this paper proposes an efficient query scheme based on the blockchain system and applies it to the data traceability scenario. The main contributions are as follows:

- To improve the query efficiency, we preprocess the data on the blockchain and design a new index structure inside the blockchain. In addition, we add a new conditional judgment field in the block header, which can effectively reduce the query overhead between query blocks.
- Compared with existing schemes, the new index structure and conditional judgement field in our proposed scheme have the lowest storage overhead. In addition, our scheme avoids

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 31,2022 at 14:34:55 UTC from IEEE Xplore. Restrictions apply.

synchronizing the data from the blockchain to an external database, which enhances data security.

• We test the proposed scheme and evaluate the results. Experimental results show that the scheme has high execution efficiency while ensuring low storage overhead.

The rest of the paper is organized as follows: Section II presents the system model. In Section III, we describe the proposed scheme in detail. In section IV we analyze the performance and security of the scheme. In section V we perform simulation experiments on the proposed scheme and analyze the results. Finally, a summary of the paper is presented.

II. System Model

In this section, we illustrate the specific functions of each entity in the scheme. As shown in Fig. 1. the system defines four participants: trusted third parties, data upload requesters, data query requesters and the blockchain system with smart contract functionality.

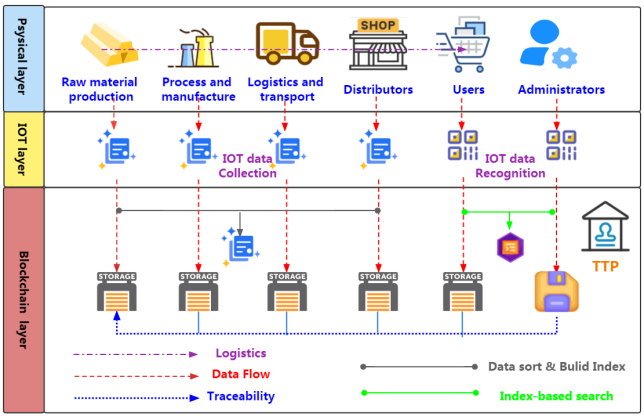

Fig. 1. System model

• **TTP**. In this scheme, the function of the trusted third party is to perform authentication and authorization for all access nodes. It is also responsible for the issuance of registration certificates as well as the renewal and revocation of certificates.

• **Data upload requestors**. In this scheme, data upload requestors can be raw material supply organizations, product processing and manufacturing organizations, logistics organizations, etc. These organizations can only access the traceability system if they have been certified by the trusted third party. As the data provider of the system, the data upload requestors can deposit product information into the system through smart contracts. When product data is stored on the blockchain, the system is able to guarantee the legitimacy of the stored data content through smart contracts.

• **Data query requestor**. In this scheme, data query requestor can be product distributors, product consumers or regulators. After authentication by trusted third parties, the enquiry node can be authorized to access the system. The query node can track and query the basic introduction of the product, product-related information and regulatory certification through the traceability system.

• **The blockchain system**. The system adopts a blockchain-based technical architecture, and each distributed upload node within the system can store data to the blockchain. Each product on the blockchain has a unique identification tag, which is a virtual identity of the actual product in the system. The tag has a unique cryptographic digital search identifier corresponding to it and has unforgeable properties.

III. Proposed Scheme

The proposed scheme is divided into five phases, including: data pre-processing, index building, access entity authorization and authentication, data upload and data search.

Phase 1. Data Pre-processing

The format of the data stored in the blockchain system can be defined as: $Y_i = < k_i, v_i >$. Y_i is a transaction tuple and each transaction tuple consists of a set of key-value relationships. The key can be defined as $key_i = (timestamp||rand_i)$, which is the unique identifier of the transaction in the blockchain generated by stitching together timestamps and random numbers. The timestamp is the time when the transaction is first submitted to the blockchain, and it should be unique and non-empty for each product. Random number $rand_i$ is a unique encrypted digital identifier generated by IOT layer related technologies, such as RFID, QR code or bar code, and converted through a certain mapping relationship. The basic storage type for $value_i$ is $json$, and each set of correspondence consists of several "fields/values". In the traceability system, we focus on the following two components: The first one is the unique identification key of the data upload entity in the blockchain, where a set of fields have been added: "ID", whose corresponding value is the unique numeric cryptographic sequence of that entity in the system. The second is the traceable identifier field. The field is "pre" which corresponds to the previous search key for this product. The pre field is generated by iteratively adding the previous transaction's search key to the current pre field. If there is no previous search key, the value of "pre" will be $null$. In other words, when the "pre" field is $null$, it means that we have searched the final source of the product. The remaining fields can be added depending on the specific application scenario that applies to the traceability system.

Phase 2. Index Building

A new index is designed in the blockchain traceability system to support fast traceability query. When a new block is ready to be created, the data inside block will no longer be changed. In this situation, the index is build by us just to optimism the query without considering operations such as adding, deleting or changing data. Existing methods of building block indexes are mainly based on B+ trees or skiplists. Both of these data structures are very efficient when querying data, but generating indexes requires additional storage space. The extra storage space increases propagation latency and consumes resources in some blockchain systems, such as Ethereum's gas. How to ensure search efficiency and

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 31,2022 at 14:34:55 UTC from IEEE Xplore. Restrictions apply.

reduce the overhead of indexing is a challenge. In this paper, we choose to create ordered arrays as indexes and use a binary search based approach as the data search algorithm. Compared to B+ trees and skip lists, sequential storage makes full use of space resources, whereas chained or tree-like structures cannot be fully utilized because they store pointers rather than real data. In addition, both require extra space to build the index structure, whereas the proposed scheme only requires sorting in raw space. The construction of the index structure is based on a fast sorting algorithm. The original chaotic data in the blockchain is efficiently rearranged in a certain order. In particular, our index structure is independent of the original blockchain structure and does not affect the transaction validation function of the original Merkle tree. The new block structure is shown as Fig. 2.

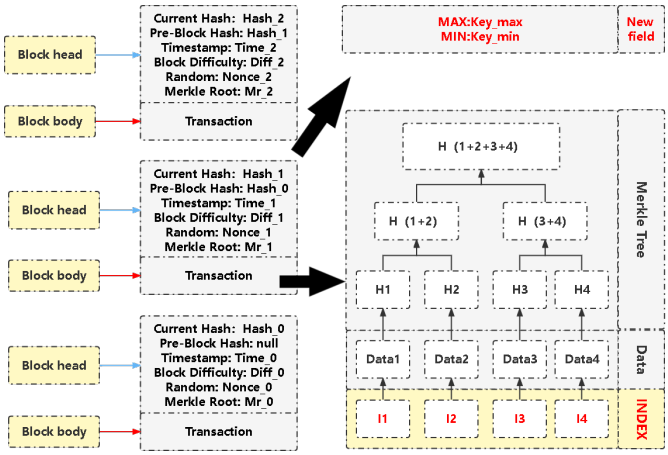

Fig. 2. The new block structure

Phase 3. Entity Authorization and Authentication

This phase should be performed by the TTP before the system can start operating. The specific functions are as follows:

(1) TTP invokes the init method to initialize the smart contract and the system.

(2) TTP can dynamically implement user information registration functions, digital certificate issuance, extension and revocation, and other functions.

Phase 4. Data Upload

This phase is performed when the data upload requestor needs to upload data to the blockchain system for storage. The process is described as follows:

(1) Upload key_i and its corresponding data text $value_i$ in the data tuple $Y_i =< k_i, v_i >$ to the blockchain for storage.

(2) Encapsulate a sorting algorithm for the orderer node, making the data in the block in a given order. When orderer node packs a block, it quickly reorders the operation objects according to the dictionary order of the timestamps. At this point, the original unordered data in the block is converted into an ordered sequence, and the orderer node generates an index based on the new sequence.

When data is uploaded to the blockchain and then packed, the miner will generate an index for the data within the block. The nature of the blockchain ensures that the new index structure in our traceability system will not be modified and queries on the index are considered trustworthy. Furthermore, the new index structure is unique and deterministic. Specifically, the value of the index is uniquely determined by the leaf nodes in the block, and Algorithm 1 guarantees that the proposed index is strictly incremental. The specific procedure for Algorithm 1 is as follows:

- *Step 1:* The object of our selection is the key of the leaf node data in the block. The system extracts the time stamp of the key and adds its maximum and minimum values as new fields to the block head.
- *Step 2:* By extracting key data from a block and reordering a new set of data in the dictionary order of the timestamp as the query index within the block.

Algorithm 1 Indexbuild

Require: The key of all leaf node data in a block;
Ensure: The sorted index sequence; max; min;

```
Sort()
for head<tial do
    return index
    if key[i]>mid then
        key[i]=key[tail],key[tail]=key[i]
        tail–
    end if
    if key[i]<mid then
        key[i]=key[head], key[head]=key[i]
        head++, i++
    end if
end for
key[head]=mid
Sort(key[ : head])
Sort(key[head+1 : ])
return max,min
```

Phase 5. Data Query

Once the data query requestor is authorized to access the system, they can search the blockchain for the required content using key-value pair relationships. Algorithm 2 is invoked by the data query requester and provides the key to be searched. The system takes the timestamp of the key as input and uses an optimized search algorithm to search for the required key based on the index value, and finally returns the query result to the data query requester. The specific procedure for Algorithm 2 is as follows:

- *Step 1:* From the latest block at the current moment, invoke each block in a loop until the Genesis block. In this paper, the block number of the Genesis block is defined as 0.
- *Step 2:* Use the target value to compare with the maximum and minimum values in the block header to determine whether the target value is within the block.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 31,2022 at 14:34:55 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Search in index

Require: The key of all leaf node data in a block key[i]; The key of the target: key[target]

Ensure: Target value successfully found: 1; Target value not found: 0.

```
for blocknum n to 0 do
    if min > key[target] or max <= key[target] then
        block−−
    end if
    if min <= key[target] <= max then
        Search( )
        if key[head] > key[tail] then
            return 0
        end if
        if middle > target then
            Search(key[head], key[middle-1], target)
        end if
        if middle < target then
            Search(key[middle+1], key[tail], target)
        end if
        if middle = target then
            return 1
        end if
    end if
end for
```

In other words, judge whether the target key satisfies $min <= target <= max$; if the condition is not satisfied, skip the block and return to step 1. If the condition is satisfied, then step 3 is executed.

- *Step 3:* When the condition 2 is reached, an optimal search algorithm is used at the index within the block to determine whether there is a target value. If the target content is found successfully, output the data corresponding to the index, and return to step 1 after completing the above steps to continue executing.

After the target key is searched, the system identifies and splits the strings in the "Pre" field to obtain the traceability information. The strings split by the algorithm are unique identifiers on the chain of previous nodes, and these strings are used as new search target key to continue the search operation in the blockchain. The complete trace chain is generated in the order of the timestamp dictionary and the trace result is returned to the data query requestor. Finally, the system generates the complete trace chain in dictionary order of timestamps and returns the trace results to the data query requestor.

IV. Theoretical Analysis

We theoretically analyze the differences between our scheme and existing mainstream works. Table 1 compares our scheme with others in five ways: whether the efficiency of data query between blocks is optimized, whether an in-block index is established, the time complexity of in-block query, the spatial complexity of the index, and whether to synchronies data to an external database. It can be seen that our system guarantees minimal overhead and query efficiency while guaranteeing secure data storage.

TABLE I
Safety and performance analysis

	[7]	[8]	[10]	[11]	[12]	Ours
IQO	No	No	Yes	No	No	Yes
BII	No	No	Yes	Yes	Yes	Yes
IQTC	-	-	$O(logn)$	$O(logn)$	$O(logn)$	$O(logn)$
ISC	-	-	$O(n)$	$O(n)$	$O(n)$	$O(1)$
DSED	Yes	Yes	No	No	No	No

IQO: Inter-block Query Optimization; BII: Build Intra-block Index; IQTC: Intra-block Query Time Complexity; ISC: Index Space Complexity; DSED: Data synchronization to external databases.

Compared with schemes [7] and [8], our scheme does not synchronize data to other external databases. The purpose of introducing blockchain technology into our traceability system is to address trust issues and prevent data that has been stored from being falsified or deleted. Synchronizing data from the blockchain system to other storage system that can not guarantee data security defeats our original purpose. Compared with other schemes that modify the blockchain storage structure and establish an index within the blockchain, our index has the lowest overhead and the highest stability.

V. Performance Evaluation

Considering the differences in blockchain development platforms and underlying functionality, A more general approach has been adopted to test the performance of our scheme. In addition, to reduce the impact of differences between blockchain systems, our system model has been simplified as much as possible, focusing more on execution efficiency. We chose two different datasets for testing. The first dataset (DS.1) is chosen as a fixed-length key-value tuple, with both the key and value data types being strings. The second dataset (DS.2) is a random length key-value tuple, with the key as a string and value as any data that satisfies the $UTF-8$ encoding rules. Each data tuple is stored on the blockchain as a complete transaction. Query efficiency is a key indicator for testing. The query method $GetHistoryForKey()$ in the fabric is chosen to be used as a baseline, which provides an API for querying the value of a specified key in a particular chain code. A timestamp is added to baseline to keep track of the time spent during the query. We compared the time consumed by the baseline and our data query methods to evaluate the performance of both data query methods.

In our experiments, to reduce the impact of remote procedure call chaining codes and storage engines on the test results, we assume that the time taken to perform processes such as query backing is the same. On the basis of the above, we test to compare the time taken by the two query methods. We created random tuples of key-values and then continued

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 31,2022 at 14:34:55 UTC from IEEE Xplore. Restrictions apply.

to post transactions until a thousand blocks of data had been generated. To make the experimental results more objective, we executed one hundred random queries. Fig. 3. shows the change in time for executing one hundred random queries as the number of blocks increases.

The query throughput is defined as the number of queries

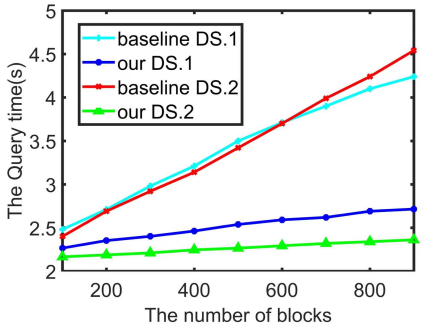

Fig. 3. Query performance with different number of blocks

that can be executed in one second. Then we compared the query throughput of the two methods. The test results are shown in Fig. 4. The experimental results show that when the number of transactions is low, the query efficiency of our solution is closer to the baseline because our solution has extra indexing overhead when storing data, which has an impact on the query efficiency when the data volume is small. However, as the number of blocks increases, the throughput of the baseline decreases more quickly. For the same number of blocks, the query throughput of our scheme is higher than that of the baseline.

According to the above experimental results, the query time

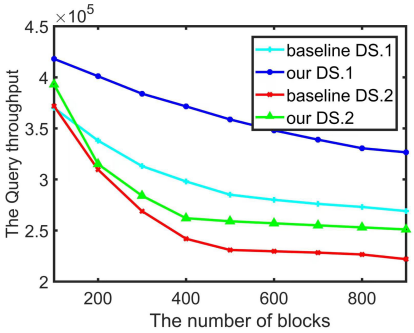

Fig. 4. The throughput of the two query methods with different block numbers

of the baseline grows faster as the number of blocks increases, while the time consumed by our solution stays at a low level. This is because the baseline needs to scan all the data each time before matching, and the query time of the baseline tends to increase linearly as the number of transactions increases. Compared to the full text scan of the baseline, the index-based query only needs to find the index location of the transaction and read the transaction data directly.

VI. Conclusion

Data traceability technology is one of the most effective ways to address product quality issues. In this paper, we proposed an optimized query scheme in blockchain-based traceability system. In our scheme, the data requester can obtain the real manufacturing process of the product, which effectively avoids the problems of incomplete data and low transparency in the traditional information traceability process. More importantly, the system can provide efficient query services while ensuring data security. Experimental results show that our scheme has higher query efficiency and lower overhead.

Acknowledgment

This work is supported by the Key Research and Development Program of Shaanxi Province under 2021ZDLGY06-02.

References

[1] Y. Cao, F. Jia, and G. Manogaran, "Efficient traceability systems of steel products using blockchain-based industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6004–6012, 2020.

[2] C. Zhuang, Q. Dai, and Y. Zhang, "Bcppt: A blockchain-based privacy-preserving and traceability identity management scheme for intellectual property," *Peer-to-Peer Networking and Applications*, pp. 1–15, 2022.

[3] X. Zhang, Y. Sun, and Y. Sun, "Research on cold chain logistics traceability system of fresh agricultural products based on blockchain," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[4] T. K. Agrawal, V. Kumar, R. Pal, L. Wang, and Y. Chen, "Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry," *Computers & industrial engineering*, vol. 154, p. 107130, 2021.

[5] X. Xu, Q. Lu, Y. Liu, L. Zhu, H. Yao, and A. V. Vasilakos, "Designing blockchain-based applications a case study for imported product traceability," *Future Generation Computer Systems*, vol. 92, no. MAR., pp. 399–406, 2019.

[6] J. Sunny, N. Undralla, and V. M. Pillai, "Supply chain transparency through blockchain-based traceability: An overview with demonstration," *Computers & Industrial Engineering*, vol. 150, no. 3, p. 106895, 2020.

[7] L. Yang, Z. Kai, Y. Ying, L. Qi, and X. Zhou, "Etherql: A query layer for blockchain system," *Springer International Publishing AG 2017*, 2017.

[8] Z. Peng, H. Wu, B. Xiao, and S. Guo, "Vql: Providing query efficiency and data authenticity in blockchain systems," in *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2019, pp. 1–6.

[9] H. Watanabe, T. Ishida, S. Ohashi, S. Fujimura, A. Nakadaira, K. Hidaka, and J. Kishigami, "Enhancing blockchain traceability with dag-based tokens," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 220–227.

[10] C. Xu, C. Zhang, and J. Xu, "vchain: Enabling verifiable boolean range queries over blockchain databases," *SIGMOD/PODS '19: International Conference on Management of Data*, 2018.

[11] P. Ruan, T. Dinh, Q. Lin, M. Zhang, G. Chen, and B. C. Ooi, "Lineagechain : a fine-grained, secure and efficient data provenance system for blockchains," *The VLDB Journal*, vol. 30, no. 1, pp. 3–24, 2021.

[12] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "Gem2-tree: A gas-efficient structure for authenticated range queries in blockchain," in *35th IEEE International Conference on Data Engineering (ICDE ?19)*, 2019.

[13] S. Wang, T. T. A. Dinh, Q. Lin, Z. Xie, M. Zhang, Q. Cai, G. Chen, W. Fu, B. C. Ooi, and P. Ruan, "Forkbase: An efficient storage engine for blockchain and forkable applications," *arXiv preprint arXiv:1802.04949*, 2018.

[14] Y. Xu, S. Zhao, L. Kong, Y. Zheng, S. Zhang, and Q. Li, "Ecbc: A high performance educational certificate blockchain with efficient query," in *International Colloquium on Theoretical Aspects of Computing*. Springer, 2017, pp. 288–304.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 31,2022 at 14:34:55 UTC from IEEE Xplore. Restrictions apply.