Information Processing and Management 58 (2021) 102518

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

A blockchain-based code copyright management system

Nan Jing [a], Qi Liu [a], Vijayan Sugumaran [b,*]

[a] SILC Business School, Shanghai University, Shanghai, P.R. China
[b] Center for Data Science and Big data Analytics, School of Business Administration, Oakland University, Rochester, Michigan, U.S.A.

A R T I C L E I N F O

Keywords:
Blockchain
Copyright management application
Node cooperation

A B S T R A C T

With the increasing number of open-source software projects, code plagiarism has become one of the threats to the software industry. However, current research on code copyright protection mostly focuses on the approach for code plagiarism detection, failing to fundamentally solve the problem of copyright confirmation and protection. This paper proposes a blockchain-based code copyright management system. Firstly, an Abstract Syntax Tree-based code originality verification model is constructed. The originality of the uploaded code is determined through its similarity to other original codes. Secondly, the Peer-to-Peer blockchain network is designed to store the copyright information of the original code. The nodes in the blockchain network can verify the originality of the code based on the code originality verification model. Then, through the construction of blocks and legitimacy validation and linking of blocks, the blockchain-based code copyright management structure is built. The whole process guarantees that the copyright information is traceable and will not be tampered with. According to the experiments, the accuracy and processing time of the code originality verification model are shown to meet code originality verification requirements. The experiment also shows that the best storage type of the code copyright information is the code fingerprint which is a 256bits hash value converted from code eigenvalues. It performs better in both response speed and storage efficiency. Moreover, because of the uniqueness and irreversibility of the result from the SHA256 algorithm, the code fingerprint storage yields a better level of storage security. In summary, this paper proposes a blockchain-based code copyright management system which provides better response speed and storage efficiency.

1. Introduction

Driven by Internet technology, open-source software development is growing exponentially. At the same time, the problem of code copyright management is becoming prominent. Firstly, the massive amount of open source code projects makes it difficult to register the authors' copyright and protection rights. Secondly, the lack of a unified code copyright management platform leads to decentralizing copyright resources and vague copyright ownership. At present, there are two main ways of handling the code copyright problem in the digital copyright field: a) perfect the construction of the relevant legal system, and b) adopt digital rights management technology based on digital watermarking (Blessing and Chakrabarti, 2009). Despite the indisputable improvements provided by these solutions, code copyright management and protection still face the following challenges:

* Corresponding author.
E-mail address: sugumara@oakland.edu (V. Sugumaran).

https://doi.org/10.1016/j.ipm.2021.102518
Received 29 April 2020; Received in revised form 9 January 2021; Accepted 11 January 2021
Available online 19 January 2021

- Because code projects are easy to duplicate and disseminate, it is challenging to obtain tort evidence, adjudicate tort behavior and defend rights by law. Therefore, legislation cannot fundamentally solve the problems existing in code copyright protection.
- The primary function of digital watermarking is to create anti-copying and anti-theft edition. However, it has not solved the copyright problem but has led to industrial monopoly and technical barriers after practical implementation.
- At the same time, most copyright management solutions are based on trusted third parties, which cannot achieve the absolute reliability of copyright information. Moreover, the above solutions' implementation cost is high, which means widespread adoption of these technologies is almost impossible.

Blockchain, which was first proposed by Nakamoto (2008)), is a decentralized ledger maintained by decentralized nodes that are utilized for distributed sharing and storing of data. Due to its properties such as decentralization, anonymity, privacy, traceability, and tamper resistance, blockchain has attracted enormous attention from scholars and practitioners (e.g., in finance, Internet of Things and medical field). Recently, Blockchain technology has been successfully applied in education credit platforms (Lin et al., 2018), threshold IoT system (Lijing et al., 2018) and privacy data preserving (Wang et al., 2018). The features of blockchain make it an innovative solution for code copyright management. The advantages of blockchain to improve the code copyright management and protection are as follows:

- Decentralization and collective verification. Decentralization is the essential feature of blockchain technology, which means that blockchain realizes distributed recording, storage, and updating of data instead of relying on centralized nodes. Thereby, the copyright management system can effectively resist DOS attacks and ensure copyright data storage security. Moreover, the decentralization mechanism means that a code project's originality needs to be verified by all record nodes before the copyright information of this code is recorded in the blockchain, which effectively avoids the low authority brought by a single decision authority.
- Transparency and openness. The data recording and updating in the blockchain network are transparent to all nodes, which is the basis for trust in the blockchain system. It provides the rights to review and trace the copyright information of original code documents. Additionally, the blockchain system is open. Blockchain nodes may freely join or leave the blockchain network. Therefore, the blockchain-based code copyright management system may conveniently attract external nodes to join the system.
- Tamper resistance and traceability. Once the information is verified and added to the blockchain, it will be stored permanently and cannot be changed. Specifically, the copyright information of the original code documents verified by the network nodes would be stored and protected by the blockchain permanently. Every node in the blockchain can also successfully query the copyright information at any time. In this way, the stability and reliability of the copyright information in the system can be significantly improved.

Based on the core advantages of blockchain technology in code copyright management, this paper proposes a blockchain-based code copyright management system. The system mainly includes two functional modules: the code plagiarism detection model that assists in document originality verification. The other is the blockchain network that realizes the packaging and storage of code documents' copyright information. In the system, any blockchain node may become one of the code authors or the miners. Furthermore, code authors can obtain a certain amount of rewards through document uploading, and miners can earn rewards according to their efforts in processing the code copyright information.

In summary, the contributions of this paper are as follows:

(1) An Abstract Syntax Tree-based code original verification model has been developed and applied in the verification process of the blockchain. By deploying automatic code eigenvalue extraction from the system server and deploying code similarity calculation on miners' local clients, the protection and verification of the original code are realized.
(2) A Peer-to-Peer blockchain network is designed to store the copyright information of original code documents. Blocks are constructed and linked by miners in the network. The copyright information in the network is permanently stored in the blockchain, making the copyright record and transfer process exhibit absolute traceability and authority.
(3) A prototype system has been implemented to evaluate the feasibility of the blockchain-based code copyright management system. It has been built upon the private blockchain based on JAVA. This system uses the transaction simulator to simulate code authors and miners uploading and verifying the code documents, and the performance of the system has been evaluated.

The remainder of the paper is organized as follows. Section 2 discusses related work and Section 3 introduces the proposed Blockchain-Based Code Copyright Management System. A scenario is set up for the experimental processing and performance evaluation, which is discussed in Section 4. Finally, a short conclusion and future work are provided in Section 5.

2. Related work

At present, the scattered copyright resources of code documents and the vague ownership of digital works, primarily of code documents, have caused a significant problem in code embezzlement and infringement of original authors' rights, which has received extensive attention from scholars in many fields. Simultaneously, blockchain technology has made rapid progress in non-monetary applications, especially in the copyright management field, because of its decentralization, tamper resistance, data traceability, and other characteristics. This section reviews the related literature in blockchain technology and its application for copyright management and code plagiarism detection.

2.1. Preliminary introduction of blockchain technologies

Blockchain originated from Bitcoin, a distributed ledger system that needs everyone to participate collectively and maintains the transcript of the entire value-transaction history (Nakamoto, 2008). The trustless and anonymous record nodes, which are called miners, maintain all the operations of the blockchain through reaching a coherent state. When a transaction occurs, the payers broadcast the transaction, and then the mine77rs gather transactions into their local blocks. A block consists of two parts: a block header and a block body. The block header contains the hash of the previous block, the timestamp of the current time, nonce, specified difficulty and Merkle root of a group of transactions. The block body contains information about the transactions. Each block in the blockchain connects to the previous block across a peer-to-peer network. Bitcoin's consistency depends on cryptographical puzzles and the Proof-of-Work concept proposed by Dwork and Naor (1993). To be specific, in Bitcoin, the validation of a block is confirmed if its hash value meets the difficulty request. Each node keeps changing the nonce and calculating hash until the difficulty request is satisfied. Once a node finds the solution to the cryptographical puzzles, its block becomes the new block and gets chained to the previous one. The process is arduous, and the winner who calculates the hash value is richly rewarded. The miners who find the solution to the puzzles will broadcast its block, including the solution, to other nodes at once. After other nodes verify the solution, the block will be accepted and added to every node's local ledgers. After finishing the process, all miners continue mining other blocks based on the new blockchain. The reward for the creator is new Bitcoins, which are created to award and stimulate miners. At the same time, all the fees for the transactions within the block are also rewarded to the creator.

O'Hara (2017) proposed the notion of the smart contract, a program code defined by users to represent a contractual agreement. Relationships between two parties in the contract are more efficiently executed using smart contracts. The smart contract and its status are stored on the blockchain and updated by record nodes. When the copyright information is uploaded and verified, the smart contract will be invoked to offer a corresponding reward. Furthermore, users can receive messages (coins or data) and send messages through a smart contract. To be specific, a smart contract comprises a storage file, program code, and account balance. Any user can send a transaction to the blockchain network and generate a smart contract which cannot be deleted or modified. Then, according to the instruction, the smart contract sends coins or data to the primary user or other users.

2.2. Blockchain for copyright management

Blockchain technology aims to realize highly reliable and traceable data management based on the distributed ledger and consensus mechanism (Moyano and Ross, 2017). The functionality of previous blockchains was limited, and lack of speed was one of the most commonly cited limitations for the application of blockchain technology. In addition, the application of blockchain should not only consider the data security problems that are a concern for people but also how to balance the speed and efficiency of the system (Henry et al., 2018). Ethereum first utilized smart contracts (chain code) into blockchain (Buterin et al., 2013), and the functions of blockchain have improved tremendously. Similarly, the application of the diversified consensus mechanism has dramatically improved the speed of block verification. Due to the above mentioned technological progress, blockchain could be used in many more situations.

The application of blockchain in non-monetary field has grown in prominence in recent years. Chen et al. (2020) proposed an incentive-aware blockchain solution called the Internet of Fake Media Things. This work provides prevention and detection of Fake News based on the advantages of blockchain technology for verification of the information being distributed. Yue et al. (2016) proposed a blockchain-based App to enable patients to own, control and share healthcare data efficiently and securely, which explores potential ways to improve existing healthcare systems while keeping patient data private. Zhao et al. (2020) constructed a privacy-preserving remote data integrity checking scheme for Internet of Things (IoT) information systems based on blockchain technologies. Baniata et al. (2021)) proposed a blockchain-assisted scheduling model for secure task assignments for cloud-based virtual reality. Oham et al. (2021)) presented a blockchain-based framework for securing smart vehicles, in which blockchain technology is applied for monitoring the internal state of vehicles and the security data exchange between vehicles. Putz et al. (2021) proposed a blockchain-based decentralized application for secure information management of Industry 4.0 assets using Digital Twins, which are the digital replica of the Industry 4.0 assets and are used by a variety of organizations across the Industry 4.0 value chain. Esposito et al. (2021) integrated identity management with authorization schemes based on blockchain technologies into a FIWARE platform (a smart city supporting platform that integrates the existing information and communications technology infrastructure) to solve data security and authority management issues in smart city applications. Berdik et al. (2021) reviewed the application of the blockchain technologies in information system applications and summarized how blockchain can improve the performance of these applications. In summary, recent applications of blockchain technologies in information system mainly focuses on information verification, data sharing, identity management and privacy-preserving. The above studies have further justified the advantages of blockchain technologies in implementing the security measures for information storage, information distribution and information sharing.

Recently, blockchain technology is being used in copyright protection and business transaction management. Savelyev (2018) studied legal-related issues of blockchain applications in copyright management and argued that blockchain could disruptively innovate the copyright management industry. Liang et al. (2020) proposed a blockchain-based Intellectual Property (IP) copyright protection algorithm that applies homomorphic encryption. Their experimental results show that blockchain technology improves the security and stability of the real-time circuit copyright authentication. Xiao et al. (2020) designed a blockchain-based IP copyright protection algorithm, which focuses on the IP circuit trading. The transaction process is improved by a distributed random embedding mechanism and position mapping function. In the field of image copyright protection, digital watermarking and perceptual hash technology can be used to generate immutable hash values, which can be loaded into the blockchain and managed through specific technology and protocol to achieve adequate protection (Meng et al., 2018). There is also recent work that explores the application of blockchain in the protection of digital music copyright. Cai (2020) combines deep learning and blockchain to construct a digital music copyright protection system and evaluate its performance.

In summary, research on blockchain applications for copyright protection is still scant, and recent works mainly focus on the fields of images, digital music and circuits copyright. Although there is no application of blockchain technology for code copyright protection, the blockchain technology has great potential to improve the state-of-the-art for code copyright protection. While the existing application of blockchain technology has proven its advantages in distributed verification and storage of data assets (Ruinian et al., 2018), it can also be applied to code plagiarism detection technology that can provide support for code originality verification.

2.3. Code plagiarism detection

Code plagiarism refers to the act of illegally copying and using other people's code for a specific purpose. There are three main types of code plagiarism detection methods: plagiarism detection based on code watermark, detection based on software birthmark, and detection based on syntax and semantics.

2.3.1. Plagiarism detection technique based on code watermark

This technique implements a unique identifier called the watermark in the code, which can identify the source code authors. That watermark can be identified or extracted by a particular extractor.

This method includes two aspects, which are static watermarking technology and dynamic watermarking technology. For the first aspect, the Davidson–Myhrvold watermarking algorithm implemented within the SAND MARK framework (DMSM), a system designed to study the effectiveness of software protection algorithms on Java bytecode (Myles et al., 2005), displays high credibility at an acceptable data-rate. However, it is vulnerable to collusion attacks, that is, an attacker can locate the watermark by making a statistical analysis of multiple watermark implantation programs. Vector Extraction Paradigm (VEP) (Stern et al., 2000) took the program as a complete statistical object, conducted statistical analysis on the instruction set to construct the eigenvector C, constructed the watermark into the same-dimensional eigenvector W, and used the instruction rearrangement to modify the program under control. However, the watermark implanted by VEP cannot be extracted. In this way, only the existence of the watermark can be verified. (Cousot and Cousot, 2004) introduced the concept of abstract software watermarking and illustrated a watermarking protocol implemented in the Java programming language.

The usage of code watermark needs to implant extra data or code in the original code document. Therefore, if that watermark is complicated, the algorithm will become too complex, which can impact the performance and stability of the program. If that watermark is simple, the concealment of it will be wholly inadequate.

2.3.2. Plagiarism detection technique based on software birthmark

This method focuses on dynamically or statically extracting unchangeable features from code as software birthmarks and then judge the similarity between the software source codes. The existence of plagiarism can be determined based on the similarity of birthmarks.

Luo et al. (2017) proposed a binary-oriented, obfuscation- resilient method based on a new concept named CoP. The CoP is represented as a series of logical formulas containing input-output relations, which utilizes the automatic theorem prover to measure the semantic similarity of two basic blocks, and then further realizes the similarity calculation of the whole software birthmark. CoP method is more resistant than other static birthmarks, but the considerable cost of symbolic execution and theoretical justification makes it challenging to handle large-scale software code. Tian et al. (2015) proposed a new type of software birthmark called Dynamic Key Instruction Sequence (DYKIS) extracted from executable code without source code. Chae et al. (2016) proposed a software birthmark named Authority Histograms (AH). Through a series of extensive experiments with different Windows applications, the paper validated both the credibility and resiliency of AH, which exceeded those of existing birthmarks. Chan et al. (2013) built the dynamic Heap Graph based Birthmark (HGB) by analyzing the dynamic heap data of the JavaScript program. Both SCDGB and HGB have the limitation of high computational complexity in a practical application.

The result of the software birthmark technique mainly relies on the high quality of the extraction. Therefore, the technology of finding and extracting the birthmark must be reliable enough to guarantee the validity of the similarity test result, which is hard to achieve.

2.3.3. Source code plagiarism detection technique

The plagiarism of source code can be divided into three types: lexical, grammatical and semantic transformation. The first two types are mainly used (Whale, 1990; Jiang et al., 2016; Verco & Wise, 1996; Roach, Gospe, Ng, & Sahin, 2014). It is a general rule that plagiarism can be judged by measuring the similarity between two code segments. Currently, attribute count and structure metrics are the two main methods used in plagiarism detection.

Halstead first proposed the attribute counting method of plagiarism detection in 1978 (Halstead, 1978). It utilized indicators of multiple measurement programs to compare programs. That may lead to a little underreporting. Although Flores et al. (2011) and Faidhi and Robinson (1987) added other measurement indexes to improve the detection efficiency, the final effect was still not ideal due to the lack of structural measurement. Therefore, a detection method based on structure metrics was proposed, which added the code's internal structure for analysis and comparison. Structure metrics are mainly divided into three types, including token, tree and graph.

For the first type, researchers translate the code into a series of tokens and then implement the comparison. Kamiya et al. (2002) proposed a new clone detection technique based on the token-by-token comparison. Cosma and Joy (2012) converted a code segment into a token sequence, and they constructed a query to obtain the distribution rule of the similarity between this code segment and all documents in the item-document matrix to evaluate the plagiarism condition of the code. Although using the token structure is resistant to code obfuscation for slight code rearrangement and variable renaming, it is still difficult to deal with complex code transformations such as redundant code implantation and control flow obfuscation.

Graph-based plagiarism detection methods are relatively few. (Liu et al., 2006) developed a PDG-based plagiarism detection tool, called GPLAG, which constructs the Program Dependency Graph (PDG) from code and implements similarity calculation through a subgraph isomorphism matching. However, its demand for the high cost of time and space makes it insufficient to cope with a mass of code plagiarism detection requirements.

Converting code into a tree structure is a more mature method in plagiarism detection analysis. Son et al. (2006) used ANTLR (a language that serves as the framework for constructing compiler and translator from grammar descriptions) to convert software code into a parse tree. They realized similarity calculation based on inner product operation and normalization of parse tree kernel. Tao et al. (2013) used the Abstract Syntax Tree (AST) of lex and yacc constructors and measured the similarity through the Hash value matching and the proportion of matched nodes. (Deqiang et al., 2017) introduced a code plagiarism detection method, named Weighted Abstract Syntax Tree Kernel (WASTK). WASTK transfers the code into an AST and gets the similarity by calculating the tree kernel of two ASTs.

2.4. Summary of the related work

In summary, most of the current research on code copyright protection focuses on improving code plagiarism detection technology, which can only complete the detection of the plagiarism code. However, it cannot fundamentally complete the verification of code copyright, prevent code embezzlement and tampering. There is no standard system for code copyright verification, protection, and trading. The successful application of blockchain technology in the non-monetary field provides a new way to solve the problems in code copyright protection. Firstly, the consensus mechanism in the blockchain network ensures sufficient verification of code copyrights. Secondly, the decentralized storage of blockchains ensures that the verified code copyright will not be tampered with after

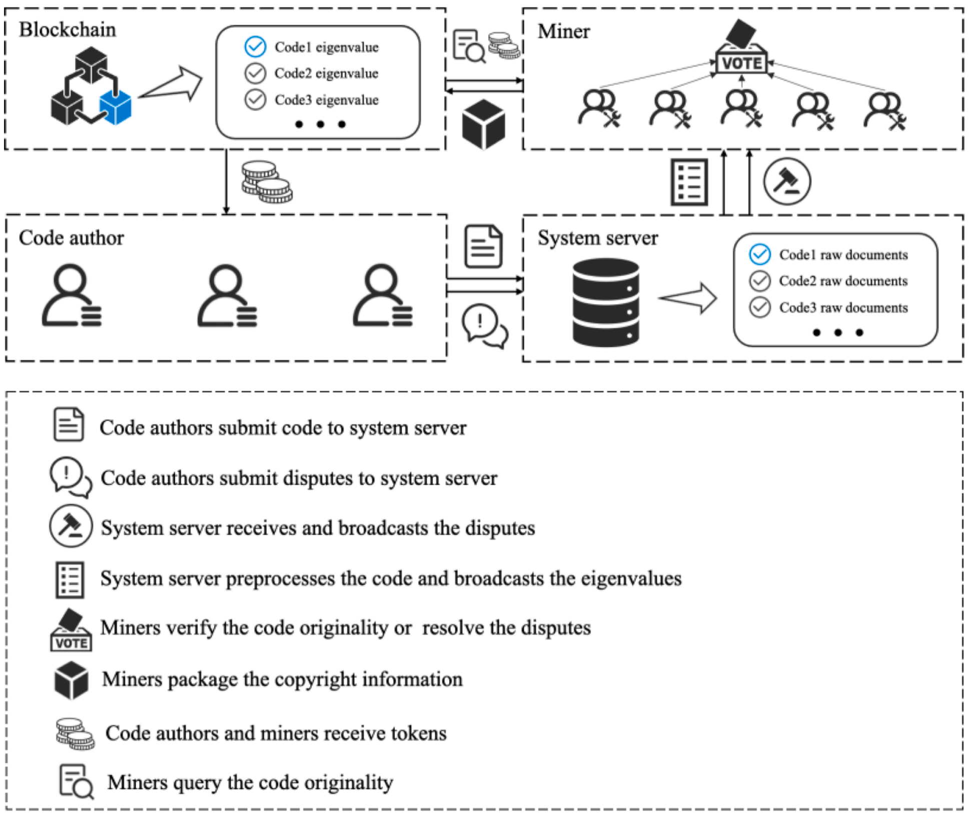

Blockchain
Code1 eigenvalue
Code2 eigenvalue
Code3 eigenvalue
Miner
VOTE
Code author
System server
Code1 raw documents
Code2 raw documents
Code3 raw documents
Code authors submit code to system server
Code authors submit disputes to system server
System server receives and broadcasts the disputes
System server preprocesses the code and broadcasts the eigenvalues
Miners verify the code originality or resolve the disputes
Miners package the copyright information
Code authors and miners receive tokens
Miners query the code originality

Fig. 1. Overview of the code copyright management system.

being stored permanently. Simultaneously, the blockchain's traceability function will meet the query requirements for the copyright of various codes. In conclusion, current research shows that the application of blockchain technology in code copyright protection has specific research significance.

3. The proposed blockchain-based code copyright management system

This section presents a detailed description of the proposed blockchain-based code copyright management system. Our basic idea is to introduce the notion of proof of data primitiveness and use the unique data structure and decentralized blockchain to develop a data preservation system. This section starts with an overview of the proposed system.

3.1. Proposed system overview

There are two types of nodes in the code copyright management system: full nodes and lightweight nodes. The full nodes maintain a complete and up-to-date copy of the blockchain. Full nodes can autonomously and authoritatively verify any transaction without external reference. The lightweight nodes are only a subset of the blockchain that submits and queries the code copyright information through full nodes. The code authors and miners can choose their node type. Firstly, the code authors will submit their code to the system server. The system server will then convert the code into code eigenvalues through preprocessing techniques such as ANTLR, and the code eigenvalues will be broadcasted across the network. The miners may verify the originality of the code, package the copyright information (which is the code eigenvalues) into blocks and link them into the blockchain. After this process, the code authors can be rewarded because of the submission of the original code. The miners can also earn a reward according to their efforts in processing the code copyright information. Every node in the network can query the copyright information recorded in the blockchain. In particular, when the submitted code contains some elements that cannot be judged by the detection algorithm, such as code reuse, or when the code author disagrees with the detection results, the system will start the manual review process. At this time, the code author needs to submit the code document together with the disputes to the system server. After the disputes are broadcasted, the miners will manually resolve the disputes based on the distributed verification technology and blockchain's consensus mechanism. Finally, the resolution will be permanently stored in the blockchain. An overview of code copyright management's working process is depicted in Fig. 1.

The current major blockchain application development platforms include Hyperledger and Ethereum. Hyperledger is mostly used for developing enterprise alliance chain, which mainly serves the internal business collaboration and can better deal with data query or modification authority issues (Nasir et al., 2018). Ethereum is a public, distributed, decentralized computing platform, mainly used to execute smart contracts (Buterin et al., 2013). Although Ethereum is a public chain, all the contracts and transactions in Ethereum run on the same main chain, which cannot meet the customization requirements. Moreover, the response speed of its main chain is often slow, which affects all applications on the Ethereum platform.

Considering the limitations of the existing platforms, we propose a novel code copyright management system that contains a specific verification mechanism to realize code originality verification. In this system, the code author will firstly upload the raw JAVA code to the code copyright verification platform. Then, the full nodes can execute an operation to extract the eigenvalues of the code. After that, the miner will verify its originality by calculating the similarity between the submitted code and other raw code stored on the blockchain. Then, the miner will encrypt it with other relevant information pertaining to the raw code document and add it to his/her hash list. Once the number of hashes stored in the local hash list reaches two, miners' host machines will run the mining program until the target block's valid hash value is found.

The key features of the proposed system are as follows:

- Decentralization. The originality of copyright is verified by the nodes in the blockchain network instead of any third-party authorities, which ensures the authenticity and reliability of copyright authentication. Moreover, the copyright information is stored on the blockchain rather than a database or cloud server to guarantee the security of data storage.
- Credibility. Before the copyright information is recorded in the blockchain, the originality of the code documents needs to be verified by the miner node based on the automatic program of code similarity calculation on the local client. Consequently, once a piece of copyright information has been recorded on the blockchain, the code's originality is credible. Moreover, due to the blockchain's tamper-resistance nature, copyright information cannot be modified or deleted once recorded on the blockchain.
- Lightweight. Most code author nodes are lightweight. Specifically, the primary demand of the code author nodes is to submit their code documents, which does not require substantial memory or computational resources. The blockchain's record nodes perform the verification computations and maintenance of the blockchain network. The lightweight nodes can submit the code documents and query the copyright information through full nodes.

3.2. Implementation of the proposed system

3.2.1. Network communications and protocol

The system is composed of full nodes and lightweight nodes. The full nodes connect each other through a credible peer-to-peer network, and the lightweight nodes connect to a certain amount of record nodes. Any code author or miner can choose the node type. Miners maintain the blockchain via a Proof-of-Work (PoW) consensus scheme and store the whole blockchain list. Time is split into several periods. During a period, copyright information on the blockchain network is collected and verified by miners. Then valid copyright information will be added and updated to the local block of individual miners. With the PoW consensus scheme, some miners' blocks will be validated and become this period's block. After that, all full nodes will update their local copy of the blockchain. The miners also start to calculate the next block.

The PoW is the original consensus algorithm in a Blockchain network. The PoW relies on a mathematical puzzle to solve for a value below a specific threshold (nonce). The first miner who finds the nonce would produce the next block and broadcast to the network. The winner is rewarded for each fixed period. According to the PoW consensus mechanism, if the miners (at least 51%) are honest, the PoW-based blockchain is non-forked. In order to realize it, PoW-based blockchain is applied to the system and assumes at least 51% of the miners are honest. As a result, once a transaction appears in the PoW-based blockchain network, it cannot be modified or deleted. Besides, assuming the encryption system hash functions are correct and ideal, no one can disobey them. Additionally, the system assumes that the user of the system is partly reliable. To be specific, their sent data should be verified, or the data is not trusted.

3.2.2. Operation mechanism of the code copyright management system

The operation mechanism of the code copyright management system mainly includes the following three steps, as shown in Fig. 2.

Step1. Code submission and preprocessing

First of all, users should register on the platform and receive initial tokens. After that, the code authors can submit the code documents through edge routers to the blockchain system server and set the transaction fee.

After receiving code documents, the system server will use ANTLR to generate Abstract Syntax Trees (ASTs) based on the raw code document. ANTLR (Terence and Russell, 1995) (Another Tool for Language Recognition) is an open-source lexical and syntax tool that can automatically generate abstract syntax trees according to the input code and visually display them.

The AST generation process could be divided into two steps:

First, after reading and analyzing the rules in the code document, ANTLR will generate a corresponding lexical analyzer and syntactical analyzer. Second, the characters in the input code will be represented by specific tokens (consisting of letters and numbers) using the lexical analyzer. The tokens will be used as input to the syntactical analyzer and then obtain the final result, namely, the AST (see Fig. 3).

Next, the AST will be optimized, linearized and converted into N-Grams

(1) AST optimization: discard the nodes reflecting headers and global variables of the code and representing variable names and constants information.
(2) AST linearization: scan and record nodes in AST through the preorder traversal to change the tree-structure into string-structure named as a token stream
(3) Get N-Grams: use a sliding window to get N-Grams from the token stream.

For the N-Gram method, Fig. 4 presents a simple example. The method users decide the size of the window (the "N" number). For a string with X tokens, the N-gram method will slide X-N+1 time and get X-N+1 Grams. As any changes in the string sequence only affect

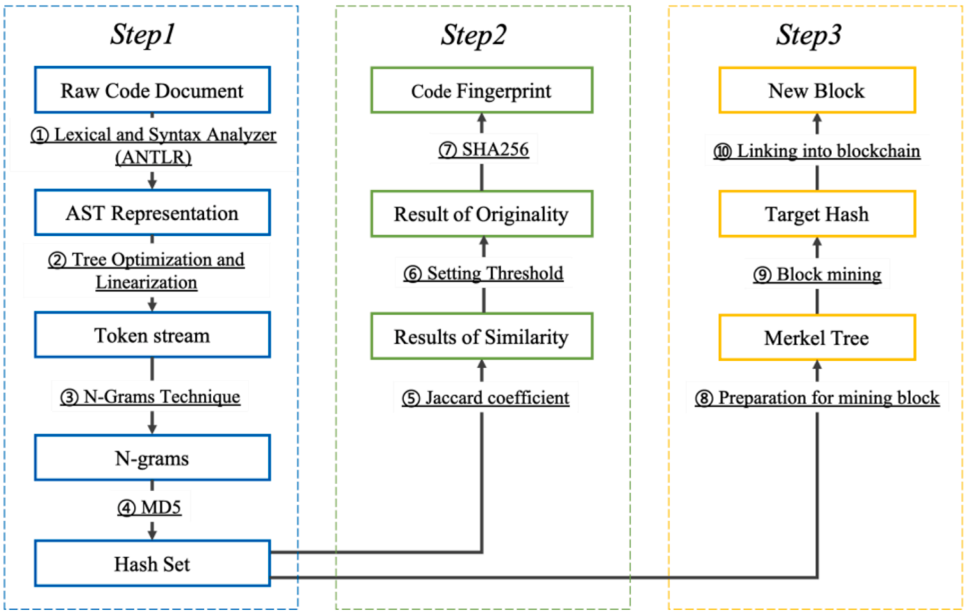


Fig. 2. The operation mechanism of the code copyright management system.

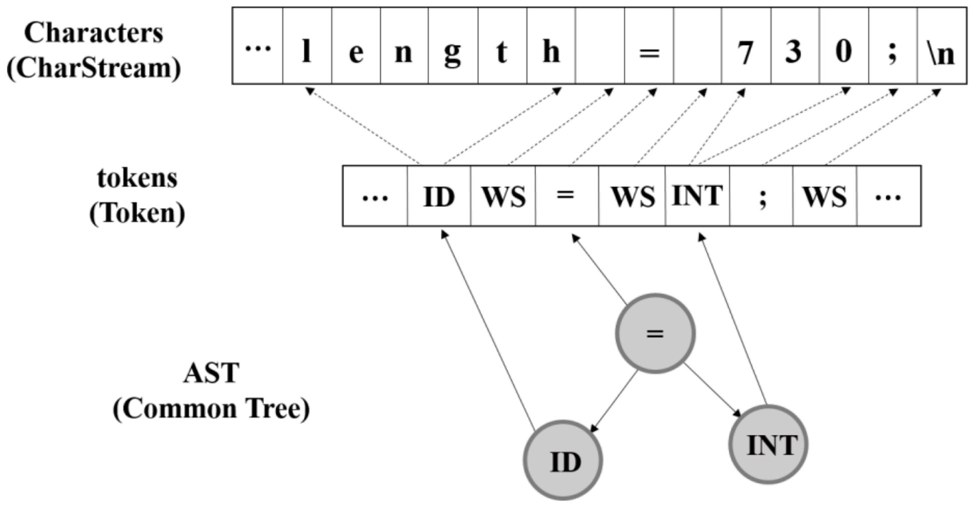


Fig. 3. ANTLR internal operation mechanism.

a few adjacent n-grams, this operation could effectively reduce the impact of code reordering and new code insertion.

The similarity calculation is based on comparing two codes' N-grams, but the length of "gram" generated by different code is different, which increases the difficulty of comparison. To improve the comparison efficiency of every "gram" pair, after obtaining the N-Grams, they are converted to a hash set by applying the MD5 message-digest Algorithm. It is a widely used cryptographic hash function, which produces a 16-byte hash value that ensures a complete and consistent information transfer. The encryption process is divided into four steps:

(1) Append the padding bits. The input message is "padded" (extended) so that its length (in bits) is equal to 448 modulo 512.
(2) Append the length. A 64-bit representation of the message length is appended to the result of step 1, which is used to record the length of the original input. The resulting message has a length that is an exact multiple of 512 bits.
(3) Initialize the MD buffer. MD5 uses four 32-bit integer parameters called link variables, which are: A= 0 × 1234567, B=0 × 89abcdef, C=0xfedcba98, D= 0 × 76543210.
(4) Start the four-round operations of the algorithm. Four buffers (A, B, C and D) are mixed with the input, using the four auxiliary functions (applying the logical operators 'and', 'or', 'not' and 'xor'). There are four rounds, and each round involves 16 basic operations.

All the N-Grams will be encrypted in this way to get a more succinct eigenvalue. At this point, the hash set of the code is successfully obtained.

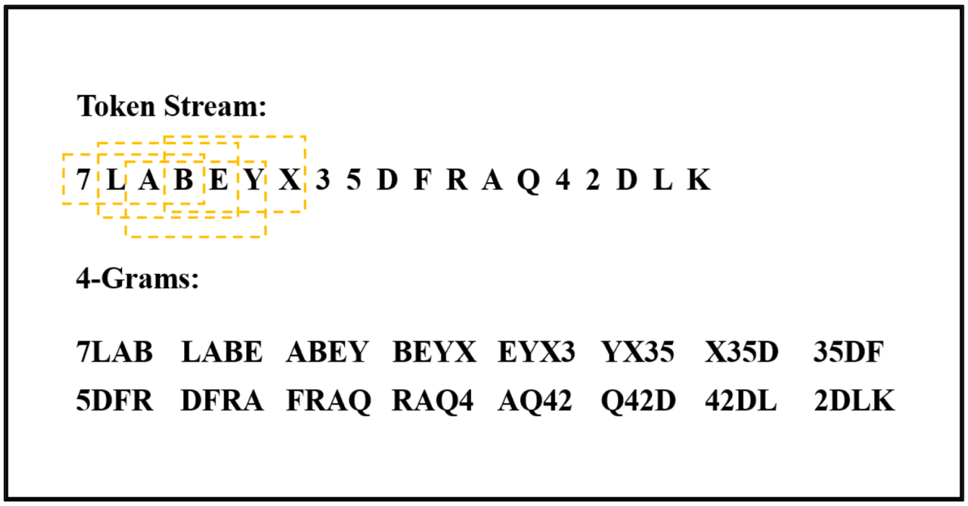

Token Stream:
7 L A B E Y X 3 5 D F R A Q 4 2 D L K
4-Grams:
7LAB LABE ABEY BEYX EYX3 YX35 X35D 35DF
5DFR DFRA FRAQ RAQ4 AQ42 Q42D 42DL 2DLK

Fig. 4. An example of a token stream converted to 4-Grams.

Step2. Consistency verifying

After getting the code's unique eigenvalue, the system can separately calculate the similarity between this code fingerprint and any other code fingerprint on the blockchain. For instance, let us assume that Cr1 and Cr2 are two codes needing similarity verification. Let SET (Cr1) and SET (Cr2) be the code hash set of Cr1 and Cr2, respectively. Then the system can evaluate the similarity score (called the Jaccard coefficient) as follows:

$$Sim(Cr1, Cr2) = \frac{|SET(Cr1) \cap SET(Cr2)|}{|SET(Cr1) \cup SET(Cr2)|} \tag{1}$$

The result of this formula will not be affected either by the different amounts between two fingerprints or by the process of generating the fingerprint. In this way, the similarity between the two programs is normalized from 0% to 100%. Then, based on a reasonable threshold, the miners can judge whether the code is original or plagiarized if the similarity score exceeds the threshold. Besides, the threshold can vary with the amount of code, which will be discussed in detail in the experimental part.

The full nodes will store the complete hash set of original code files in their block. In order to improve storage efficiency, the lightweight nodes will store the code fingerprint of the original code files, which is the SHA 256 digest of the hash set. SHA256 is also one of the cryptographic hash functions. It generates an almost unique, fixed size 256-bit (32-byte) hash.

The preceding steps indicate the overall process of transforming the original code into 256bits hash values. It can be summarized as follows. After obtaining the n-grams from the raw code, the n-grams will be encrypted by the MD5 algorithm which can retain the code's information to a large extent and get the corresponding MD5 hash set. After that, we will concatenate the hash set as a single string and use the SHA256 algorithm to encrypt the string, which will attain the final result called code fingerprint in our system.

It should be noted that the system will start the manual review process when the code contains certain elements that cannot be analyzed by the detection algorithm, such as code reuse. In that situation, the elements of the code will be extracted by the system server. Then, the extracted part will be verified manually by the miners, and the remaining code will still be checked by the detection algorithm. The final plagiarism detection results will be determined by the combination of manual detection and algorithmic detection.

The manual review will also be applied when the code author disagrees with the detection results. If the author insists that the code is original, yet the system detection result shows the code is plagiarized, the code author needs to submit the code together with the description of the disputes to the system server. Then the miners will resolve the disputes manually after the description of disputes is broadcasted. Based on the distributed verification technology and the blockchain consensus mechanism, the system will give the final arbitration result. Suppose the dispute is caused by the plagiarized code that was added into the copyright management system earlier than the original one, in which case, the previous plagiarized code will also be removed from the system database. Finally, the arbitration result will be stored in the blockchain permanently.

Step3 Code copyright recording and token receiving

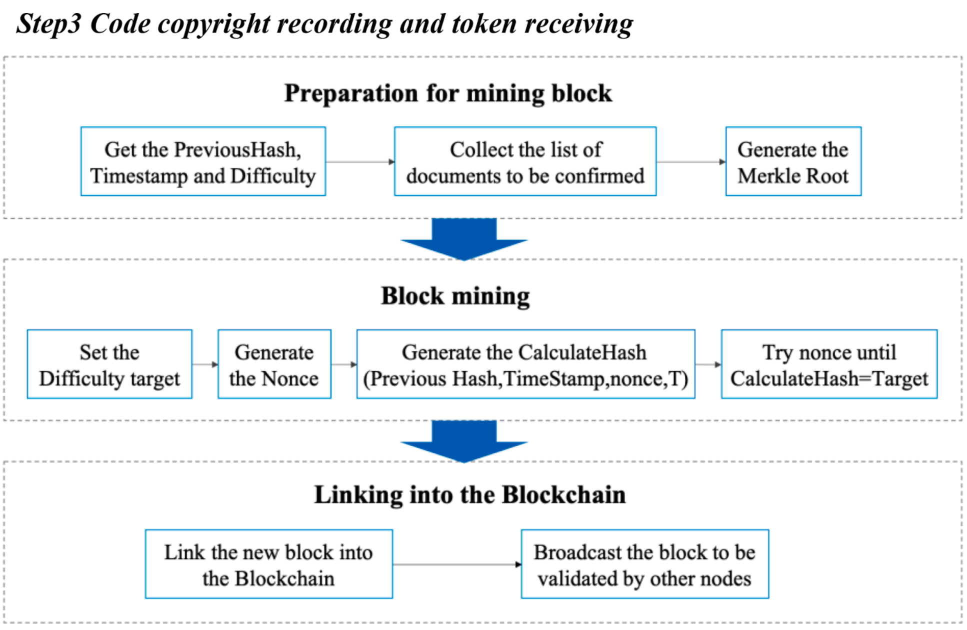

Step3 Code copyright recording and token receiving
Preparation for mining block
Get the PreviousHash, Timestamp and Difficulty
Collect the list of documents to be confirmed
Generate the Merkle Root
Block mining
Set the Difficulty target
Generate the Nonce
Generate the CalculateHash (Previous Hash,TimeStamp,nonce,T)
Try nonce until CalculateHash=Target
Linking into the Blockchain
Link the new block into the Blockchain
Broadcast the block to be validated by other nodes

Fig. 5. The process of generating a new block.

In this work, the code copyright recording and reward receiving process employs the PoW mechanism, which includes three steps: preparation for mining block, block mining, and linking into the Blockchain, as shown in Fig. 5. The specifics of the mechanism are briefly described below.

First, before the miners in the blockchain compete for the recording right of the next block, they need to synchronize the latest information of the current blockchain, and obtain the hash value of the previous block, difficulty value and timestamp. Miners also need to collect the original code copyright information in the local code pool during the latest period, and then generate the target hash value based on all the information above.

After data preparation, miners start a competition for recording rights of the new block by solving the cryptographic puzzle. The cryptographic puzzle is to identify the value of the nonce. A nonce is a random number which is added to the contents of a block. By adding nonce, the hash output of the data will change. The first miner who finds the hash of the data, which is less than a preset number, wins block recoding rights.

Algorithm 1 shows the pseudo-code of calculateHash(), which calculates the block hash value with the current nonce. This function invokes the SHA256 digest algorithm to generate a hash value with the current nonce to identify the nonce's right value.

Algorithm 2 shows the pseudo-code of mineBlock(), which finds the hash meeting the difficulty requirement. This function is invoked when the miners mine the block (i.e., compete for the recording right of the next block). First, the target hash is set based on the difficulty value. The nonce is accumulated from 0, and the calculateHash() function is called to calculate the hash value with the current nonce until the current hash that matches the target hash is found.

At this time, the first miner who successfully solves the puzzle will generate a block and push that block into the network for verification from other nodes. Other nodes can verify the validity of the block information and nonce value. The blocks that meet the conditions can be successfully linked into the blockchain.

Algorithm 3 shows the pseudo-code of isBlockValid(), which verifies the validity of the block. This function checks the block's validity from three perspectives: whether the current hash value is correct, whether the previous block connected by the current block is correct, and whether the hash value of the current block matches with the difficulty requirements. Only if all three conditions are met, the current block is valid. Then, the valid block is permanently stored in the distributed ledger and queried by all nodes.

3.3. *The contribution of the proposed system*

Our blockchain-based code copyright management system fundamentally realizes the verification, storage, and inquiry of code copyright. First, the application of blockchain technology achieves the efficiency and reliability of code copyright management. Based on the consensus mechanism of blockchain technology, the system ensures that the originality of code is protected by every node of the whole blockchain network, avoiding any decision made by a single decision authority. Second, the security of data storage is significantly enhanced. Based on the decentralized storage mechanism of the blockchain, the copyright information of the original code can be stored in the blockchain permanently, which avoids copyright tampering and embezzlement. Finally, blockchain technology better ensures the transparency and openness of copyright management. Based on blockchains' traceability, all the copyright information of verified original code can be queried through the proposed system. In conclusion, blockchain technology and its application in our proposed system fit the needs of copyright management concerning credibility and security, and also improves the performance of the copyright management process.

4. Experimental processing and performance evaluation

This section describes the experimental performance evaluation of the blockchain-based code copyright management system by constructing the system based on JAVA. The efficiency of the blockchain-based code copyright management system mainly depends on the performance of the code originality verification model and the blockchain platform. Therefore, this validation is carried out concerning the following two aspects. The first part is the performance of the code originality detection model. The corresponding performance evaluation of the JAVA-based code copyright management system is evaluated in the second part. The processing time and storage size are used to assess the efficiency of storage. The next sub-section describes the prototype system setting.

4.1. *Prototype system setting*

This paper utilizes JAVA-based blockchain as the platform. The prototype system is implemented using laptops and virtual

Algorithm 1
calculateHash() for calculating the hash value with a current nonce.

```
Input:
  previousHash: the hash value of the previous block
  timeStamp: current timestamp
  nonce: a random number
  data: the original code copyright information to be stored
Output: the hash value with the current nonce
1: call calculatedhash←applySHA256(previousHash, timestamp, nonce, data) // calculate SHA256
2: return calculatedhash
```

Algorithm 2
mineBlock() for finding target hash.

```
Input: difficulty
Output: block hash matching the target hash
1: set_target(difficulty) // set target hash based on the difficulty
2: while currentHash ≠ target hash
3:         nonce++
4:         current_hash←calculateHash()
5: end while
5: return current_hash
```

Algorithm 3
isBlockValid() for verifying the validity of the block.

```
Input:
currentHash: the hash value of the current block
previousHash: the hash value of the previous block
currentBlock: the information of the current block (previousHash, timestamp, nonce, data)
previousBlock: the information of the previous block (previousHash, timestamp, nonce, data)
Output: whether the current block is valid
1:  if currentHash ≠ calculateHash(currentBlock)
2:    return false
3:      else
4:          if previousHash ≠ previousHash in previouseBlock
5:              return false
6:        else
7:            if currentHash ≠ target hash
8:                return false
9:          else
10:             return true
11: end if
```

machines. Specifically, the laptops used had the following configuration: Intel i7-7820HQ CPU at 2.90 GHz, 24 GB of memory, and Windows 10 Education. The blockchain is deployed locally (Buterin et al., 2013). The process of code uploading and validating is simulated locally, through which the local code files can be stored in the blockchain. In the experiment, local code files are stored in the blockchain to evaluate the mechanism's performance.

4.2. Code originality verification model performance

In the blockchain-based code copyright management system, the originality of the submitted code files needs to be verified before the copyright information is appended to the blockchain. In the proposed system, the verification process is based on the code originality verification model.

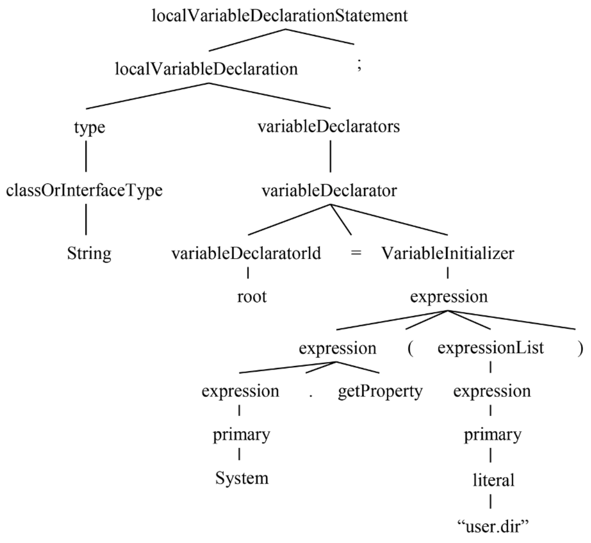


Fig. 6. A part of the generated AST.

In order to ensure the effectiveness of the model in different situations and improve the applicability of the model, the experiment selects two data sets of JAVA source code files. The size of each code in the two data set is different. Each code file of the first data set contains about 60 lines, and the other one contains about 200 lines of code on average. Each data set includes 51 JAVA source code files, and one of them is set as the original program. Twenty-five documents in the test set are modified intentionally based on the original programs and imitating the common plagiarism methods, such as complete copy, marker renaming, single statement or code block rearrangement, adding or deleting annotations and changing control logic. The rest of the test set is JAVA code files which are not related to the original program document.

4.2.1. *AST generation and linearization*

First, the 51 raw code files in each dataset are transformed into ASTs (Abstract Syntax Tree) through ANTLR. Fig. 6 presents a partial AST generated by the original code.

Then, we use a tool from ANTLR named "walker" to scan and record nodes in AST through the preorder traversal. Table 1 gives some examples of token streams transformed by two JAVA code files.

4.2.2. *Model optimization*

Next, we can use the optimization process to decide whether a code file has been plagiarized. First, we use the N-grams algorithm to divide the token stream into short N-grams. After that, the N-grams will be encrypted by the MD5 encryption algorithm. Then we use the Jaccard algorithm to calculate the similarity between two code files. Finally, whether the code is plagiarized is judged according to the similarity threshold, which is determined by the training results of the code plagiarism detection model. In the entire process, the parameter N of the N-grams algorithm and the similarity threshold have a significant impact on the originality judgment result. The value of N represents the granularity of code detection, which determines the extent that the code is subdivided. The similarity threshold represents the upper limit of similarity for judging plagiarism, which determines the rigor of code detection. Therefore, before doing the above process, we should set the appropriate value for N and similarity threshold.

(1) Evaluation indices

To evaluate which N value and similarity threshold are the most suitable for the system, we should firstly choose the evaluation indices. In this experiment, we qualify the system's performance with a particular N value using precision and recall (Saxena et al., 2020; Singh et al., 2020; Raghavan et al., 1989; El-Alami et al., 2020). In this system, precision is the fraction of correctly classified positives among the classified positives, while recall is the fraction of correctly classified positives retrieved over the total number of actual positives.

$$precision(P) = \frac{correctly\,classified\,positives}{classified\,positives} \tag{2}$$

$$recall = \frac{correctly\,classified\,positives}{actual\,positives} \tag{3}$$

(2) N value and similarity threshold confirmation

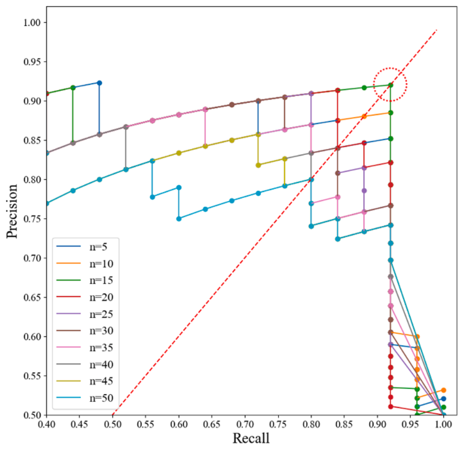
We conducted experiments on two datasets. For each data set, we first select a series of N values and then measure the precision and recall when the similarity threshold is changed. Finally, according to different similarity thresholds at each N value, the precision-recall curve is shown in Fig. 7.

To better measure the location of the optimal recall and precision point, the y=x trend line is introduced in Fig. 7, which is the red dotted line in Fig. 7(a) and 7(b). The closer the point is to the trend line, the better the recall and precision are. In other words, the N value and similarity threshold of the point closest to the trend line are optimal.

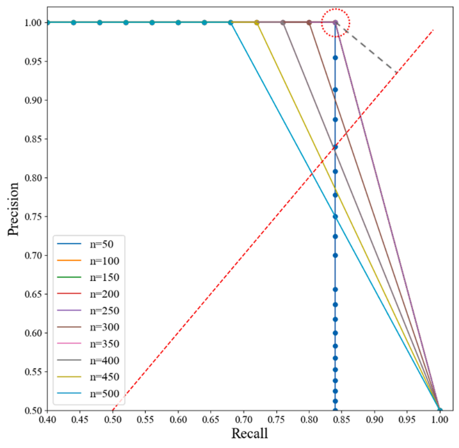
For the data set with an average code size of 60 lines, we set the value range of N as 5 to 50 with an interval of 5. It can be seen in Fig. 7(a), when N=15, the overall precision and recall ratios are higher, and when similarity threshold value=0.375, which is the point marked by the red circle, the recall and precision of the detection algorithm are both 0.92, and the performance of the algorithm is optimal.

Table 1
Token stream.

Code Files	Token Stream
Code1. Java	CompilationUnit PackageDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName ImportDeclaration QualifiedName TypeDeclaration ClassOrInterfaceModifier……
Code2. Java	LocalVariableDeclarationStatement LocalVariableDeclaration Type ClassOrInterfaceType VariableDeclarators VariableDeclarator VariableDeclaratorId VariableInitializer Expression Creator CreatedName ClassCreatorRest Arguments ExpressionList Expression Primary BlockStatement Statement StatementExpression Expression Expression Expression Primary ExpressionList Expression Primary Expression Primary BlockStatement Statement……



(a) The Precision-Recall curve of the first data set



(b) The Precision-Recall curve of the second data set

Fig. 7. The Precision-Recall curve in the two data sets.

For the data set with an average code size of 200 lines, considering the apparent increase of code size, we set the value range of N as 50 to 500 with 50 as the interval. Fig. 7(b) shows that the optimal precision-recall curve includes multiple curves such as N = 250. In order to show the optimal results more clearly, we list the N values of all the optimal curves and the optimal similarity threshold of each N value in Table 2. The optimal precision and recall are also listed in the table.

According to Table 2, with the four optimal parameter combinations, the detection algorithm can obtain the optimal precision 1.0 and optimal recall 0.84 on the data set with an average code size of 200 lines.

Through analyzing the experimental results of the two datasets, we can draw the following conclusions. First of all, there can be various combination of N values and similarity threshold parameter to obtain the optimal detection algorithm. When the interval of the selected N value is reduced (such as reducing the N value interval of the first experiment from 5 to 1), more optimal parameter pairs can be found according to the experiment result. Another finding is that when the code size increases, the optimal N value will increase obviously compared with the smaller code because the lower detection granularity may fail to meet the requirements due to the significant increase in the number of characters in the code. Simultaneously, the similarity threshold will be reduced correspondingly, considering that the more extensive code often contains other elements such as configuration or locale, which will reduce the proportion of plagiarism code. Therefore, the code copyright management system needs to adjust the parameter combinations dynamically based on the code size to achieve dynamic optimization.

4.2.3. *Plagiarism judgment system test*

Using the plagiarism judgment system with the confirmed parameter (15-grams and similarity threshold = 0.375), we carried out plagiarism judgments on two new JAVA code samples with an average size of 60 lines. One of them is plagiarized, while the other is the original code. Furthermore, these judgments are based on ten original certified code files on the blockchain (OC3-OC12). Both of these two new code files will be compared with each of the ten certified code files individually. The results of the comparison are shown in Figs. 8 and 9.

Fig. 8 shows that part of the similarity values between new code1 and other certified code files exceed 37.5%, which means that new code1 has plagiarism. However, we can see from Fig. 9 that all judgment results of new code2 are less than 37.5%. Then, new code2 will be recognized as the original code and will finally be added to the blockchain. Thus, the experiment described above reveals that this system can accomplish the task of detecting code plagiarism effectively. After this initial test, we chose 15 new source code files and tested them using the plagiarism judgment system. The results showed that 10 of them are original source code files and would be prepared for storing on the blockchain.

4.3. *JAVA-based code copyright management system characteristics*

Now we examine the impact of storage on the performance of the proposed blockchain-based code copyright management system

Table 2
Optimal parameter combinations for the second data set.

Optimal N value	Optimal similarity threshold	Optimal precision	Optimal recall
100	0.1251	1.0	0.84
150	0.0932		
200	0.0392		
250	0.0120		

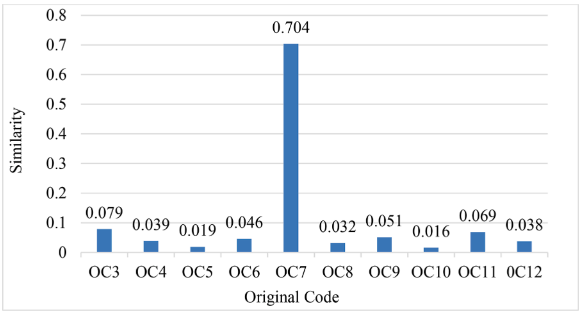


Fig. 8. The comparison result of new code1.

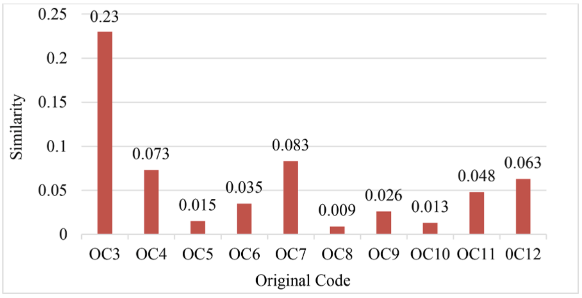


Fig. 9. The comparison result of new code2.

based on JAVA. Although there is no definite limit on the storage size in a block, as the amount of copyright information stored in the block increases, the data storage content will significantly impact the blockchain's size and the whole copyright management system's response time. So, this section first compares the storage efficiency and operation efficiency of two storage formats. Ten different-size code files are selected as test data. As shown in Fig. 10, the code fingerprint and the hash set have similar up trends with respect to time, mainly because as the number of code files increases, code documents need to be queued for processing. However, the fingerprint mode has a little better performance in terms of time consumed, and the advantage will be more evident as the amount of code increases gradually. Therefore, the storage based on code fingerprint can provide a faster response and enhance the system's efficiency in the size test.

The result of storage efficiency between the code fingerprint and the hash set is depicted in Fig. 11 and Fig. 12. As shown in Fig. 11, when the amount of stored code copyright information increases, the storage size has an apparent increase for the stored data regardless of whether the type is hash set or code fingerprint. However, since the code fingerprint is the result of the SHA256 algorithm and has a fixed length, the code fingerprint of each code file has the same storage size. Therefore, the growth of the storage size is steady. In contrast, the storage size of the hash set depends on the content of the code, the growth tendency of the hash set storage type fluctuates wildly and has no pattern.

Fig. 12 reflects the difference in the storage size between the two data types of storage. Since code fingerprint is the SHA256 summary of the code's hash set, its storage volume is compressed. The average size of the code fingerprint only accounts for 282.9 bytes, but the hash set's average size is 10446.1 bytes. Therefore, the code fingerprint data type improves the storage efficiency of the blockchain.

It can be seen from Figs. 10, 11, and 12 that the fingerprint storage mode has better performance for both time and size. Specifically, the code fingerprint storage type has a quicker response time with less information storage size. At the same time, because of the uniqueness and irreversibility of the result from the SHA256 algorithm, code fingerprint can be verified by any of the nodes in the blockchain, which can prevent the copyright information from potential intrusion and ensure the integrity and security of the blockchain network. Thus, the fingerprint type is sufficient to represent the copyright characteristics of the current code files and our system adopts the fingerprint storage type.

4.4. *Discussion*

The blockchain-based code copyright management system has significant advantages compared with the central database-based system. The blockchain consist of a chain of blocks, and each block's hash is partly decided by the previous hash and the data stored in the block. Once the data stored in the block changes, its calculation also cannot match the prior hash. With these features, the blockchain can provide the necessary functions for preventing tampering and data traceability. Thus, the blockchain-based copyright management system is useful for detecting plagiarism, such as complete copy, modification or deletion of annotations, adjustment of

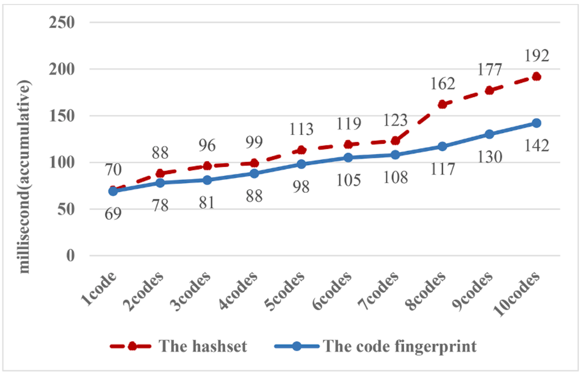


Fig. 10. Differences in response time in two data types of storage.

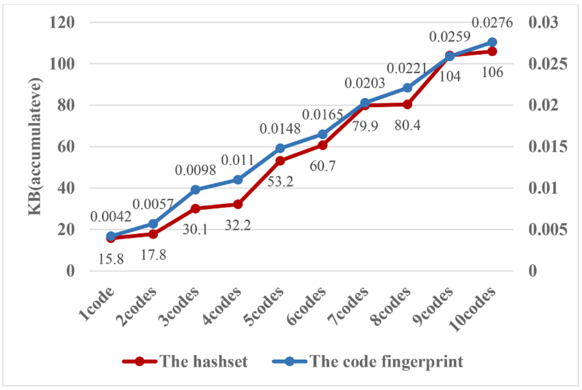


Fig. 11. Differences in storage size of the two data types (line chart).

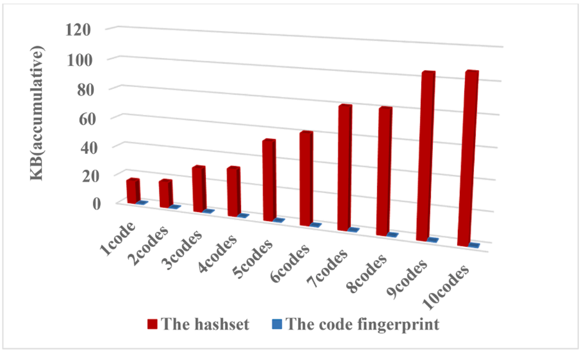


Fig. 12. Differences in storage size of the two data types (bar chart).

code block order and controlling logic. It is exceedingly important because it can ensure the authenticity of code copyright.

Another advantage of the blockchain-based code copyright management system is its response time and processing speed, which have been demonstrated in this work. Compared to the hash set storage mode, the code fingerprint storage type has a quicker response time with less information storage size. Thus, using the code fingerprint-based storage type can enhance the storage efficiency of the entire blockchain network. Furthermore, when the number of code files awaiting validation increases, the blockchain-based system shows better performance concerning response time (based on the average response time) for the main steps.

The functionality implementation and system efficiency mainly depend on the type of the adopted blockchain platform. As mentioned in Section 3.1, the existing platform is not applicable to the proposed system regarding blockchain type and response efficiency. Therefore, we have developed a new system based on the JAVA platform. On the one hand, the system development based on the JAVA platform can choose the type of blockchain freely and has fewer restrictions without relying on the existing development framework. On the other hand, using the JAVA platform, the system can be deployed locally, making it more convenient for observing experiment results.

5. Conclusion

In this paper, we have proposed a blockchain-based code copyright management system. This system consists of full nodes and lightweight nodes. Each author or miner can choose the node type. Firstly, the authors submit the code and set the transaction fee. After generating the code files' eigenvalue, the full nodes will broadcast the code copyright information to the miners. The miners can then verify the originality of the code files by calculating the similarity between the target code and the chained code files. The code whose similarity with all chained code files is lower than the predefined similarity threshold can be marked as original code files. The original code's copyright information includes the code fingerprint, ID, similarity result, and author's name. Moreover, all of this information will be recorded in a block and linked to the blockchain. After the block is verified, the miners can receive rewards. As such, the system supports the permanent storage and protection of the original source code files. In addition, the stored data on the blockchain can be queried by all of the nodes in the proposed system.

The proposed system applies blockchain technology for improving the safety and reliability of the code copyright process compared to the traditional copyright management system. This decentralized system is more credible, as every node in the system has the right to verify the code copyright rather than relying on the judgment from a third-party authority. Besides, the code's similarity is calculated by the miners enabled by an automatic program, and the final code copyright information is recorded on the immutable blockchain. These procedures ensure the credibility of the blockchain-based system. Moreover, the lightweight nodes can inquire about the historical data of the full nodes instead of storing all information on the blockchain, which not only reduces the resource consumption but also improves the response time.

Our future work will focus on optimizing the techniques for extracting eigenvalues and calculating the similarity and the process for copyright originality verification. This paper puts forward a viable solution to resolve the disputes in code originality judgment and copyright protection. Future work will further evaluate the performance of these functionalities in the proposed code copyright management system, especially for manually resolving the copyright disputes. Although the blockchain structure can improve the credibility of manual detection, it should be noted that this method may still have limitations in practice, such as operational efficiency. The future work will also extend and improve the manual detection method to reduce its limitations in terms of operational efficiency. Meanwhile, in the proposed system, the copyright trading function will be implemented based on the smart contract technology to achieve flexible and controllable copyright transactions. Moreover, by changing the eigenvalue extraction process and the similarity calculation algorithm, the platform can be applied for copyright protection of different types of documents, such as images, texts, and digital music. Besides, the system can also be applied to various situations where reliable verification against the digital form of assets is required. As the blockchain system is traceable and unmodifiable, it can also be used to verify the authenticity of invoices, discount coupons, or other electronic documents.

CRediT authorship contribution statement

Nan Jing: Supervision, Conceptualization, Methodology, Writing - review & editing. **Qi Liu:** Conceptualization, Methodology, Data curation, Software, Validation, Writing - original draft. **Vijayan Sugumaran:** Methodology, Validation, Writing - review & editing.

References

Baniata, H., Anaqreh, A., & Kertesz, A. (2021). PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. *Information Processing & Management, 58* (1), Article 102393.

Berdik, D., Otoum, S., Schmidt, N., Porter, D., & Jararweh, Y. (2021). A survey on blockchain for information systems management and security. *Information Processing & Management, 58*(1), Article 102397.

Blessing, L. T. M., & Chakrabarti, A. (2009). Drm, *a design research methodology*. DOI: 10.1007/978-1-84882-587-1.

Buterin, V., Wiederhold, B., Riva, G., & Graffigna, G. (2013). A next-generation smart contract and decentralized application platform. In *Ethereum Whitepaper*.

Cai, Z. (2020). Usage of Deep Learning and Blockchain in Compilation and Copyright Protection of Digital Music. *IEEE Access, 8*, 164144–164154.

Chae, D. K., Ha, J., Kim, S. W., Kang, B. J., Im, E. G., & Park, S. J. (2016). Credible, resilient, and scalable detection of software plagiarism using authority histograms. *Knowledge-Based Systems, 95*(Mar.1), 114–124.

Chan, P. P. F., Hui, L. C. K., & Yiu, S. M (2013). Heap graph based software theft detection. *IEEE Transactions on Information Forensics and Security, 8*(1), 101–110.

Chen, Q., Srivastava, G., Parizi, R. M., Aloqaily, M., & Al Ridhawi, I. (2020). An incentive-aware blockchain-based solution for internet of fake media things. *Information Processing & Management*, Article 102370.

Cosma, G., & Joy, M. (2012). An approach to source-code plagiarism detection and investigation using latent semantic analysis. *IEEE Transactions on Computers, 61*(3), 379–394.

Cousot, P., & Cousot, R. (2004). An abstract interpretation-based framework for software watermarking. *A.C.M, 39*, 173–185.

Deqiang, F., Yanyan, X., Haoran, Y., & Boyang, Y. (2017). Wastk: a weighted abstract syntax tree kernel method for source code plagiarism detection. *Scientific Programming*, 1–8.

Dwork, C., & Naor, M. (1993). Pricing via processing or combatting junk mail. In *International Cryptology Conference on Advances in Cryptology*.

El-Alami, F. Z., El Alaoui, S. Q., & En-Nahnahi, N. (2020). Deep Neural Models and Retrofitting for Arabic Text Categorization. *International Journal of Intelligent Information Technologies, 16*(2), 74–86.

Esposito, C., Ficco, M., & Gupta, B. B. (2021). Blockchain-based authentication and authorization for smart city applications. *Information Processing & Management, 58* (2), Article 102468.

Faidhi, J. A. W., & Robinson, S. K (1987). An empirical approach for detecting program similarity and plagi*arism within a university programming environment. Computers & Education, 11*(1), 11–19.

Flores, E., Barron-Cedeno, A., Rosso, P., & Moeno, L. (2011). Towards the detection of cross-language source code reuse. *Lecture Notes in Computer Science, 6716*, 250–253.

Halstead, M. H. (1978). *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc.
Henry, R., Herzberg, A., & Kate, A. (2018). Blockchain access privacy: challenges and directions. *IEEE Security & Privacy, 16*(4), 38–45.
Jiang, M., Zhang, F. F., Wu, D. H., Liu, P., & Zhu, S. C. (2016). Deviation-based obfuscation-resilient program equivalence checking with application to software plagiarism detection. *IEEE Transactions on Reliability, 65*(4), 1–18.
Kamiya, T., Kusumoto, S., & Inoue, K. (2002). Ccfinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering, 28*(7), 654–670.
Liang, W., Zhang, D., Lei, X., Tang, M., Li, K. C., & Zomaya, A. (2020). Circuit Copyright Blockchain: Blockchain-based Homomorphic Encryption for I.P. Circuit Protection. *IEEE Transactions on Emerging Topics in Computing*.
Lijing, Z., Licheng, W., Yiru, S., & Pin, L. (2018). Beekeeper: a blockchain-based iot system with secure storage and homomorphic computation. *IEEE Access*, 1.
Lin, C., He, D., Huang, X. Y., Choo, K. R., & Vasilakos, A. V. (2018). BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of Network and Computer Applications*, 42–52.
Liu, C., Chen, C., Han, J., & Yu, P. S. (2006). GPLAG: Detection of software plagiarism by program dependence graph analysis. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* A.C.M.. August 20-23, 2006.
Luo, L., Ming, J., Wu, D., Liu, P., & Zhu, S. (2017). Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection. *IEEE Transactions on Software Engineering, 43*, 1157–1177.
Meng, Z., Morizumi, T., Miyata, S., & Kinoshita, H. (2018). Design scheme of copyright management system based on digital watermarking and blockchain. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*.
Moyano, J. P., & Ross, O. (2017). KYC optimization using distributed ledger technology. *Business & Information Systems Engineering, 59*(6), 411–423.
Myles, G., Collberg, C., Heidepriem, Z., & Navabi, A. (2005). The evaluation of two software watermarking algorithms. *Software: Practice and Experience, 35*(10), 16.
Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system.*
Nasir, Q, Qasse I, A, Abu Talib, M, et al. (2018). Performance analysis of hyperledger fabric platforms[J]. *Security and Communication Networks*, 2018.
O'Hara, K. (2017). Smart contracts - dumb idea. *IEEE Internet Computing, 21*(2), 97–101.
Oham, C., Michelin, R. A., Jurdak, R., Kanhere, S. S., & Jha, S. (2021). B-FERL: Blockchain based framework for securing smart vehicles. *Information Processing & Management, 58*(1), Article 102426.
Putz, B., Dietz, M., Empl, P., & Pernul, G. (2021). EtherTwin: Blockchain-based Secure Digital Twin Information Management. *Information Processing & Management, 58*(1), Article 102425.
Raghavan, Vijay, Bollmann, Peter, & Jung, Gwang S (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM. Transactions on Information Systems, 7*(3), 205–229.
Roach, E. S., Gospe, S., Ng, Y., & Sahin, M. (2014). Trust but verify: the introduction of plagiarism detection software. *Pediatric Neurology, 50*(4), 287.
Ruinian, L., Tianyi, S., Bo, M., Hong, L., Xiuzhen, C., & Limin, S. (2018). Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 1.
Savelyev, A. (2018). Copyright in the blockchain era: Promises and challenges. *Computer Law & Security Review, 34*(3), 550–561.
Saxena, R, Adate, A. S., & Sasikumar, D. (2020). A Comparative Study on Adversarial Noise Generation for Single Image Classification. *International Journal of Intelligent Information Technologies, 16*(1), 75–87.
Singh, A., Nagaraj, N., Hiriyannaiah, S., & Patnaik, L. M. (2020). ISCG: An Intelligent Sensing and Caption Generation System for Object Detection and Captioning Using Deep Learning. *International Journal of Intelligent Information Technologies, 16*(4), 51–67.
Son, J. W., Park, S. B., & Park, S. Y. (2006). Program plagiarism detection using parse tree kernels. *PRICAI 2006: Trends in Artificial Intelligence*. In *9th Pacific Rim International Conference on Artificial Intelligence*.
Stern, J., Quisquater, J. J., Hachez, G., & Koeune, F. (2000). Robust object watermarking: application to code. *Lecture Notes in Computer Science, 1768*, 368–378.
Tao, G., Guowei, D., Hu, Q., & Baojiang, C. (2013). Improved plagiarism detection algorithm based on abstract syntax tree. In *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*.
Terence, P., & Russell, Q. (1995). Antlr: a predicated-LL(k) parser generator. *Software Practice & Experience, 25*(7), 789–810.
Tian, Z. Z., Zheng, Q. H., Liu, T., Fan, M., Zhuang, E. Y., & Yang, Z. J. (2015). Software plagiarism detection with birthmarks based on dynamic key instruction sequences. *IEEE Transactions on Software Engineering, 41*(12), 1217–1235.
Verco, K. L., & Wise, M. J. (1996). Plagiarism a la mode: a comparison of automated systems for detecting suspected plagiarism. *The Computer Journal, 39*(9), 741–750.
Wang, J., Li, M., He, Y., Li, H., Xiao, K., & Wang, C. (2018). A blockchain based privacy-preserving incentive mechanism in crowdsensing applications. *IEEE Access*, 99, 1-1.
Whale, G. (1990). Software metrics and plagiarism detection. *Journal of Systems and Software, 13*(2), 131–138.
Xiao, L., Huang, W., Xie, Y., Xiao, W., & Li, K. C. (2020). A Blockchain-Based Traceable I.P. Copyright Protection Algorithm. *IEEE Access, 8*, 49532–49542.
Yue, X., Wang, H., Jin, D., Li, M., & Jiang, W. (2016). Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of Medical Systems, 40*(10).
Zhao, Q., Chen, S., Liu, Z., Baker, T., & Zhang, Y. (2020). Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems. *Information Processing & Management, 57*(6), Article 102355.