Hindawi
Security and Communication Networks
Volume 2021, Article ID 9927640, 17 pages
https://doi.org/10.1155/2021/9927640

Research Article

A Blockchain-Based Editorial Management System

Eman-Yaser Daraghmi,[1] **Mamoun Abu Helou**,[2] **and Yousef-Awwad Daraghmi**[3]

[1]*Department of Applied Computing, College of Applied Sciences, Palestine Technical University-Kadoorie, Tulkarm, State of Palestine*
[2]*Department of Management Information System, Al Istiqlal University, Jericho, State of Palestine*
[3]*Department of Computer Systems Engineering, Palestine Technical University-Kadoorie, Tulkarm, State of Palestine*

Correspondence should be addressed to Eman-Yaser Daraghmi; e.daraghmi@ptuk.edu.ps

Received 22 March 2021; Accepted 27 April 2021; Published 6 May 2021

Academic Editor: omar cheikhrouhou

Research publications are reaching a stunning growth rate. Therefore, new challenges regarding managing the peer-review activities are presented, such as data security, privacy, integrity, fragmentation, and isolation. Further, because of the emergence of predatory journals and research fraud, there is a need to assess the quality of the peer-review process. This research proposes a fully functional blockchain-based editorial management system, namely, TimedChain, for managing the peer-review process from submission to publication. TimedChain provides secure, interoperable, transparent, and efficient access to manuscripts by publishers, authors, readers, and other third parties. Time-based smart contracts and advanced encryption techniques are employed for governing transactions, controlling access, and providing further security. An incentive mechanism that evaluates publishers' value respecting their efforts at managing and maintaining research data and creating new blocks is introduced. Extensive experiments were conducted for performance evaluation. Results demonstrate the efficiency of the proposed system in governing a large set of data at low latency.

1. Introduction

Nowadays, the technology of blockchain goes far beyond its primary use. It is adopted worldwide to improve the management of information assets owned by organizations or individuals. The blockchain provides the infrastructure in order to simplify collecting, managing, preserving, storing, and delivering information. Thus, it can be employed in various fields, such as health records management, academia, Internet of Things (IoT) data, digital voting, research publications, and many more. It allows information to be available to the right users at the right time.

Research publications are reaching a stunning growth rate which increases annually by 8–9%. This growth is a result of the increasing number of researchers and the hypercompetitive environment of science. Before publishing a research paper, it undergoes a long process, namely, peer-review, to assess its quality before publication. The process of peer-review means the scrutiny and the evaluation of an author's research by other experts in the same field before publication. This process is managed by the editorial management system (EMS) which employs the Information and Communication Technologies (ICT) to facilitate the overseeing of all editorial content and the process of peer-review. EMS is the support tool that manages the peer-review process and enhances its quality, facilitates the editorial tasks (i.e., manuscript submission, tracking its progress from submission to publication), improves the decision-making processes, and enhances the communication between authors, editors, and reviewers. Currently, several commercial and open source editorial management systems are available and in use, such as ScholarOne [1], Elsevier [2], Aries [3], Bepress [4], Open Journal Systems (OJS) [5], and eJManager [6]. These systems vary in their offered features and services to improve the publisher's productivity and archive the historical data preserved by the publisher, such as review reports, contact information, and manuscript versions [7].

Currently, there is a rapid increase in the number of publications, the options to conduct scholarly publishing, and the demand to publish. Thus, new challenges are introduced, including ensuring the quality of peer-review, preventing plagiarism, avoiding predatory publishing, ensuring the security and privacy of scientific data, improving the academic integrity, managing the coordinated data, data fragmentation and isolation, effective data sharing, interoperability access, and preventing fraud [8]. Blockchain is one of the most powerful technologies that has the potential to address these challenges. Few blockchain-based editorial management platforms have been proposed, such as Pluto [9], ARTiFACTS [10], ScienceMatters-EUREKA [11], and Orvium [12]. Further discussion will be illustrated in Section 2.2.

This research proposes a fully functional blockchain-based editorial management system, namely, TimedChain. The proposed system is designed to manage the editorial activities and the peer-review process from submission to publication. TimedChain provides secure, interoperable, transparent, and efficient access to manuscripts by publishers, editors, authors, readers, and other third parties. In this work, time-based smart contracts that meet the demands of peer-review are proposed. These contracts are embedded in the blockchain to govern and control transactions, as well as to monitor the actions carried out on scientific data (i.e., manuscripts, review reports, review scores, etc.). The proposed smart contracts enforce acceptable usage policies to manage all the editorial activities including manuscript submission and handling, peer-review processes, and final decision making. To improve the security of the proposed system, advanced encryption schemas and techniques are also adopted.

Additionally, instead of rewarding publishers with cryptocurrencies, this work introduces a Proof of Authority based incentive mechanism which estimates and evaluates the value of all publishers in the network. The proposed mechanism estimates the publishers' efforts at managing publications, maintaining scientific data, and creating new blocks. A publisher who has the least value will be chosen for creating the new block. An incentive will be rewarded to the "block's creator" publisher and added to its value to minimize the possibility of recreating the next block, hence reaching fairness status and achieving system's sustainability. The proposed TimedChain system was tested through extensive experiments, and the results demonstrate its efficiency in governing a large set of data at low latency.

2. Background and Related Work

2.1. Background. In 2008, Nakamoto [13] introduced the technology of blockchain as the core of the Bitcoin cryptocurrency (i.e., digital currency). A blockchain is defined as a decentralized and distributed peer-to-peer network in which each performed transaction by any blockchain's participant is placed in a public immutable single ledger. Each block within the blockchain network includes (1) a block's header that lists a value representing the hash of the prior block, (2) a Merkle root, (3) a timestamp, and (4) a data part that includes data related to transactions.

To identify the blockchain's participants, public-private key cryptography is utilized [14, 15]. For public identification, a public key is employed, whereas for authorizing transactions, the private key is employed. In fact, any transaction in the network includes (1) the sender's public key, (2) data, and (3) the hash of the former transaction. Because of the data part, a blockchain can maintain several digital assets as logs, certificates, records, transcripts, licenses, and property rights. Blocks are connected in order in accordance with their hash values. In the blockchain network, the chain of blocks is duplicated over the distributed blockchain network and stored by minors.

To manage the relations among the blockchain participants, smart contracts are utilized. Smart contracts are the digital mean or the computer codes that organize, secure, and formalize relations among participants over the blockchain network. It is executed and run over all the nodes within the blockchain network [5] to govern, manage, and control the transactions [6]. In order to decrease and reduce the malicious breach, smart contracts may embed various types of collateral, bonding, contractual clauses, and property rights in computer software or hardware [16]. Smart contracts enable trusted transactions and agreements to be performed among anonymous and distinct entities (i.e., the network nodes) without the need for an external enforcement schema or a central authority. Smart contracts have been implemented by several blockchain-based projects, such as the Ethereum platform [17] and Hyperledger [18], that enable creating dynamic and scalable rules, policies, conditions, and terms to securely share and exchange data.

For managing the mining process, different types of proof are used. The Proof of Stake (PoS) and the Proof of Work (PoW) are consensus algorithms that are utilized in blockchain-based frameworks to determine the miner's block to be appended next. In 2017, Gavin Wood who is the cofounder of the Ethereum platform and former CTO introduced the Proof of Authority (PoA) consensus algorithm as a replacement for the PoS and the PoW [19]. To set a private blockchain-based network, the PoA considers the value of the nodes in order to define the set of "block validators" or "authorities." Authorities are arbitrarily selected as trustworthy entities to create the new blocks and secure the blockchain network. They are staking their reputation instead of coins to maintain security. Accordingly, improving the system's security, maintaining its privacy, lowering the time-consuming computations, and increasing the performance of the system by requiring minimum latency in accepting transactions and stable time periods for issuing blocks are the benefits of the PoA.

2.2. Related Work: Blockchain-Based Editorial Management Systems. Despite the fact that the technology of blockchain was first introduced to present the Bitcoin cryptocurrency, spreading its nonfinancial utilization is an objective of researchers.

Previous research shows that a blockchain technology has the capabilities to impact the process of managing scientific data and extend its potential. Blockchain technology not only can enhance the management of data, but also can improve the default auditability as all database access operations will be verifiable. Research in this field is quite new but growing rapidly.

The authors in [8] present a governance framework for scientific publishing. They employ the model of a consortium-based blockchain for efficient publishing process. In [9], Pluto uses smart contracts to build a decentralized publishing system that allows users to submit scientific information and retain copyright control. Pluto proposes the concept of "reputation score" which is calculated based on research contributions and peer-review process. The ARTiFACTS system in [10] uses blockchain as underlying technology to record scholarly artifacts in immutable chains so that research outputs can be captured and shared easily. To enable the integration with open access research repositories, the ARTiFACTS focuses on handling only the process of research creation, paper tracking, and output sharing. Similarly, EUREKA in [11] utilizes smart contract and tokens for rating paper review. In this system, authors pay review fees which are converted to tokens and rewarded to reviewers. The Orvium publishing platform in [12] aims at making the review process open and transparent, reducing publication cost, and rewarding reviewers with Orvium tokens. Further, Orvium enables institutions to create and manage decentralized journals in the Orvium environment.

In [20], a decentralized publication system for open science is proposed. The proposal aims to challenge the technical infrastructure that supports the middleman role of the oligopoly of traditional publishers. In [21], the authors propose a decentralized blockchain-based solution for managing scientific communication to solve the challenges and the incentive problems of traditional systems. The authors in [22] employed a smart contract on the blockchain to present the implementation of a modifiable research paper. They proposed a decentralized scholarly communication platform using blockchain network. In [23], a blockchain-based system for scientific knowledge with a community management framework, namely, Aletheia, is proposed. It is designed to host peer-to-peer information allowing people to transfer scientific journals to one another, with the community being facilitated by smart contracts. It is hoped these smart contracts could form the basis for administering other open source software projects.

2.3. Contributions. Building upon previous efforts, this research proposes a fully functional blockchain-based editorial management system, namely, TimedChain, to manage the editorial activities and the peer-review process from submission to publication. The proposed system provides secure, interoperable, transparent, and efficient access to manuscripts by publishers, editors, authors, readers, and other third parties.

Unlike previously proposed systems, TimedChain is designed to be built above the current databases stored by publishers. The blockchain of the TimedChain system stores the hashes of the data references while sending the actual link for queries in a private transaction over HTTPS.

Moreover, TimedChain employs the proxy reencryption and the deposit-box techniques, which allows storing keys and small encrypted records directly on the blockchain, thus improving the transfer of records to other third parties. Furthermore, timed-based smart contracts are adopted to manage access to accounts for the varying roles of publishers, editors, authors, readers, and other third parties on the blockchain. This allows for a stratification of roles that can suit the various needs of users. Additionally, the design of the TimedChain system is based on a consensus algorithm which plays an important role in improving the validation process when adding new nodes to the network or removing harmful users. Finally, instead of rewarding publishers with cryptocurrencies, this research introduces a Proof of Authority based incentive mechanism which estimates and evaluates the value of all publishers in the network. The proposed mechanism estimates the publishers' efforts at managing publications, maintaining scientific data, and creating new blocks. A publisher who has the least value will be chosen for creating the new block. An incentive will be rewarded to the "block's creator" publisher and added to its value to minimize the possibility of recreating the next block, hence reaching fairness status and achieving system's sustainability.

3. TimedChain Architecture

3.1. System Overview. Figure 1 shows the architecture of the proposed system which facilitates the interaction between the nodes (publishers, authors, reviewers, researchers, and so on). A "third-party-typeA" node represents a reviewer, while a "third-party-typeB" node represents a researcher who would like to access the published manuscripts preserved and archived by the publisher.

To facilitate the integration with publishers, TimedChain is designed to be built above the current databases stored by publishers. Accessing scientific data will be performed through the smart contracts stored in the blockchain. Accordingly, the blockchain will maintain the history of this access carried out on the scientific data, hence meeting data integrity and preventing misuse of it.

A novel incentive mechanism that is combined with the consensus algorithm, namely, Proof of Authority (PoA), is adopted in this work. It evaluates the value of publishers from the perspective of EMSs by estimating the efforts of publishers regarding maintaining scientific data and generating new blocks. The value of a publisher node specifies its significance regarding the quality and quantity of scientific data stored in its database. Four major indicators, namely, Correctness, Consistency, Completeness (3C's), and nonredundancy, are used to evaluate the quality of scientific data stored in publishers' databases.

The proposed incentive mechanism implies that each publisher node will be categorized as a block creator or a

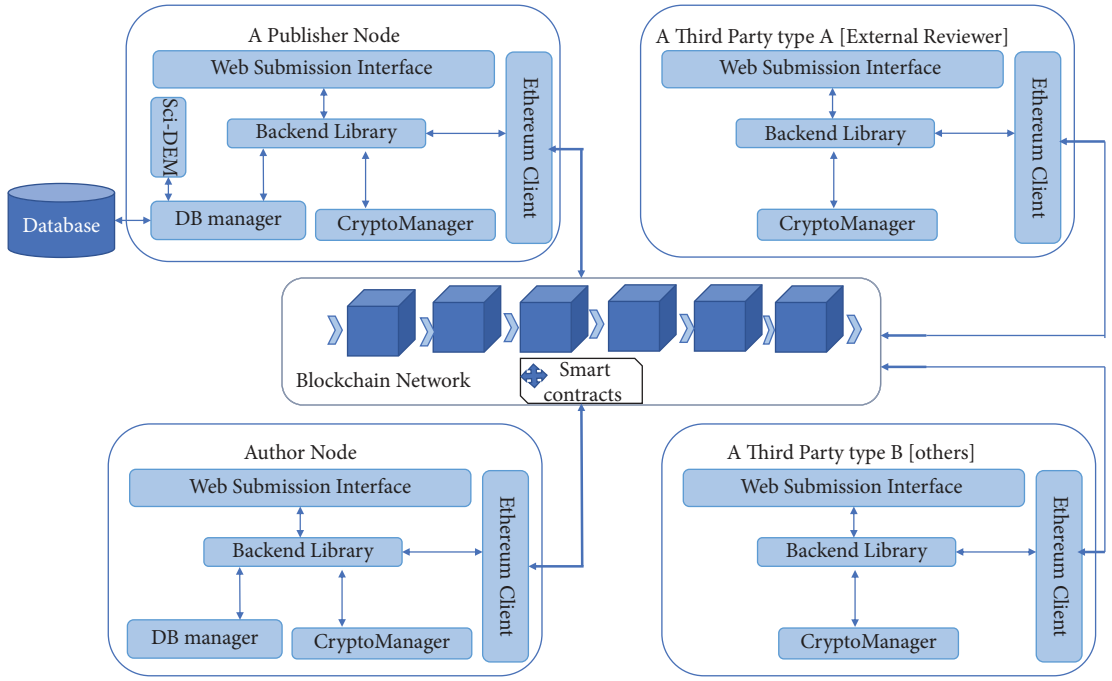

Figure 1: The components of the TimedChain system.

voter node regarding its value. A "block creator" is a publisher node that has the lowest value among all other publisher nodes. Any node that has a value greater than or equal to the average value of the TimedChain network will be classified as a "voter node." Voter nodes validate the IDs of new nodes that intend to join the system. This minimizes the possibility of adding illegitimate nodes to system. A block creator node will be rewarded an incentive that will be added to its current value to lower its probability of recreating the next block rather than rewarding it a digital currency, thereby maintaining the sustainability of the system and reaching fairness status. Moreover, improving the quality of scientific data can be considered as another advantage of the proposed incentive mechanism since publishers will compete to maintain and/or create more correct, consist, complete, and unique scientific records to increase their values. This decreases their probability of performing the computational task of creating the new next block.

3.2. TimedChain Software Components. This section demonstrates the main components of the TimedChain system, as shown in Figure 1.

3.2.1. Scientific Data Evaluation Manager (Sci-DEM). It is a python-based tool that evaluates the value of publishers during the initialization stage or when adding/editing scientific data stored in the publisher's database. The value of each node demonstrates the significance of a publisher regarding the quantity and quality of the maintained scientific data. Extracting features, managing related data, and organizing the unstructured parts are the main functions of the Sci-DEM. A classification technique with semantic, syntactical, and lexical analysis of the record is adopted [24]. The value of each publisher node will be calculated by (3) and stored in the Publishers Contract (PC) to classify nodes as (1) the voters' nodes and (2) the creator of the next block.

3.2.2. DB Manager (DBM). The DB manager is a Golang API that is designed to control all access to publishers' database, and it is managed by the blockchain's permissions. The DB manager stores scientific data, navigates the existing database, and ensures data integrity. When submitting a new manuscript, the DB manager calculates the hash of the manuscript and its access links to be stored in the Records Contract (RC). Additionally, it calculates the hash of each log to be stored in the Mining Contract (MC).

3.2.3. CryptoManager. This component employs the encryption/decryption schemas. According to the proposed design, three encryption schemas are employed: (1) symmetric key encryption; (2) public key encryption; (3) proxy reencryption. The scientific data is stored and encrypted using a symmetric key encryption schema. The CryptoManager generates a symmetric key, which is reencrypted using the public keys of the author node, publisher node, and proxy node. Further, the public key encryption schema is used to provide a secure distribution of the information among nodes over HTTPS.

In addition, the temporary records saved in the Deposit-Box Contract (DBC) are also encrypted using the public key encryption schema; these records are used to facilitate the access by a "third-party-typeB" node, while the proxy reencryption schema is adopted to facilitate the access by a "third-party-typeA" node.

3.2.4. Ethereum Client (Eth.Client). Joining the blockchain network requires an access point. The Eth.Client component includes all the required functionalities to allow a node accessing and joining the Ethereum blockchain network [25]. TimedChain is a permissioned blockchain system. Thus, via the Eth.Client, a node that has a permission can access the blockchain network. The Go Ethereum client, which employs JSON RPC endpoints on the Internet, is used for implementation [26]. The Go Ethereum client allows accessing the information of a node over HTTPS via the use of wallets which in turn function differently based on the type of the node.

3.2.5. Web Submission Interface. The web submission component enables managing the editorial activities by providing the entire required functions from manuscript submission to publishing.

3.2.6. The Backend Library. To enable the Eth.Client component to communicate with the blockchain and the embedded smart contracts, the backend library performs low-level parsing and formatting, as well as exporting a function call API to abstract the communication with the blockchain.

3.3. Smart Contracts. They are implemented in blockchain-based systems to monitor, manage, and govern transactions. The proposed smart contracts are shown in Figure 2. According to the smart contracts' design shown in Figure 2, a set of timers (i.e., timing functions) and connectivity operations are employed to ensure that transactions are performed in acceptable time, and an authorized transaction is intended. Next, we provide a detail for each part of the smart contracts which are deployed in TimedChain.

3.3.1. Nodes Contract (NC). This contract is a general contract that keeps the Eth.Add of all nodes registered in the system to facilitate all functions provided by the system and to avoid double registration.

3.3.2. Publishers Contract (PC). PC is a contract that preserves the data of publishers' nodes (i.e., publishers) to organize the procedures of mining, the activities of registration, and certain overwrite functions for the blockchain. It maps the identification string (ID) of a publisher node with its Ethereum address (Eth.Add); i.e., the Eth.Add of a node is considered as its public key.

PC maintains the value of all publisher nodes in the system to determine "voters' nodes" and "a block creator node." A voter node is a publisher node that has a value which is equal or greater than the average value of system. A creator node is a publisher node that has the lowest value in the system. Thus, PC has two Boolean data fields labeled as "voter" and "creator." The PC applies a combination of both the Proof of Authority (PoA) consensus algorithm [27] and the new proposed incentive mechanism proposed for mining. PC maps the Eth.Add of a publisher node with its corresponding RHC.Add.

PC exploits its coded policies, procedures, and rules to regulate adding and registering new IDs. Publishers who are labeled as "voters' nodes" validate any other node that requests a new higher-level role. The set of "voters' nodes" will secure the system against any potential threats. To initialize the system, initially the PC will first be empty with no registered nodes. After that, the system will add an administrative node (i.e., temporary node) which acts as an initial publisher node. Later on, once enough nodes join the system, the temporary node is deleted and removed from the network.

Moreover, the PC applies the overwriting procedures, such as leaving the system because of going out of business or removing those nodes which may damage the system by revoking and canceling their permissions. First, a request of "removing a node" will be submitted by a voter node. Once the request reaches the majority of votes, the node's type will be overwritten as terminated. The node will be removed from the PC, and all of its related information from the various contracts will be deleted.

3.3.3. Authors Contract (AC). AC is a contract that preserves the data of authors (i.e., authors). Similar to the PC, AC maps the identification string (ID) of an author node to its associated Eth.Add and RHC.Add.

3.3.4. Third Parties Contract (TPC). TPC is a contract that preserves the data of third parties (i.e., external reviewers, editors, and researchers who would like to collect data). Similar to the PC and AC, TPC maps the ID of a third-party node to its associated Eth.Add. The proposed system employs two techniques to access data via an authorized third party: proxy reencryption and deposit-box.

3.3.5. Relation History Contract (RHC). RHC preserves the history of the relationship of each node in the system or a summary list of the relationship where the academic research data is stored and managed by publishers' nodes (peer-reviewed journals/conferences). The RHC of an author's node maintains references to all peer-review publishers that have connection with that author. The RHC of a publisher node has references to all authors' nodes that have submitted research. The RHC will be created for every publisher or author node through a node's registration.

As shown in Figure 2, the RHC is recognized by the Eth.Add of the RHC's owner. Each row in the RHC lists the Eth.Add of an engaged node, its IDs, and status (i.e., newly established, pending, approved, declined). The main advantage of using the status data field ID is to allow and enable notifications. The RHC row also uses the timestamp for the date field to specify the time of the latest updates of the status field, the timer field, and the RC.Add.

3.3.6. Records Contract (RC). RC tracks all research data (i.e., records) stored by peer-reviewed journals/conferences.

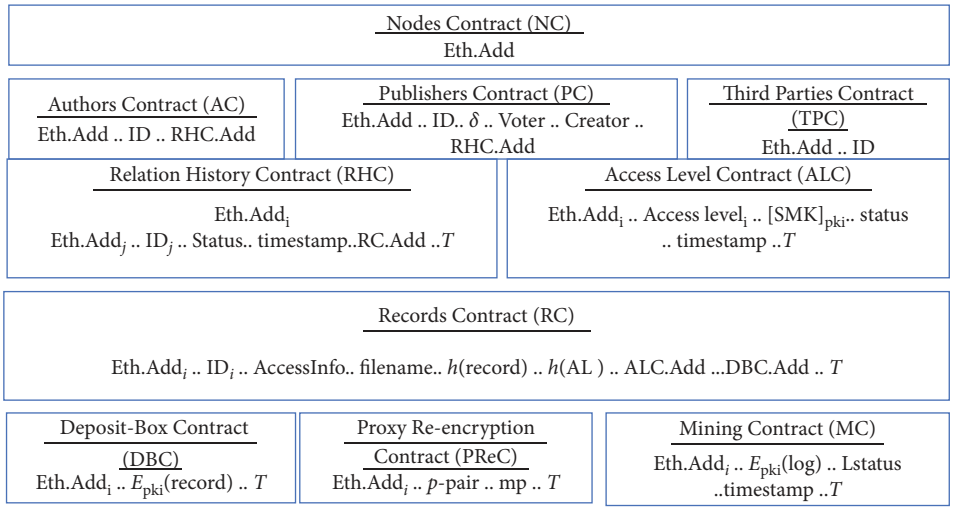

Figure 2: The proposed smart contracts.

The RC is created when establishing a new relation between publisher and author nodes.

The Eth.Add of the owner node is used to identify the RC. Each row in the RC includes two data fields: a filename which specifies the record ID and the access info which provides the required information to access the database of a publisher node. Additionally, to ensure the integrity of data, a hash value of the record's access link and a hash value of the record itself will be stored. Moreover, the RC lists both the ACC.Add and the DBC.Add.

3.3.7. Access Control Contract (ACC). The details of all permissions are listed in this contract. It lists the Eth.Add of nodes which have access permissions. This contract includes the following data fields: the "AccessLevel," an encrypted symmetric key, status, timestamp, and a timer T.

The AccessLevel field demonstrates the level of access of the node that has the listed Eth.Add, such as owner, read/edit, read only, blind-read, and temp-read. The owner level is assigned to the publisher node that creates the access link for the record and adds it to the database. This level means that a node has a complete control on the ACC. Owner node has permission to (1) add other nodes with the other level, (2) alter the level of existing nodes, and (3) remove nodes from the ACC.

A node with "read/edit" access level can read and/or edit the record. (1) This node has the symmetric key which is generated for encrypting the record when the record was created, or (2) the node gets the symmetric key through proxy reencryption.

A node with "temp-read" level can temporarily read a record as it has the DBC.Add. A node with "read only" access level can only read the record. The "blind-read" access level demonstrates that the PReC can retrieve the symmetric key encrypted for the proxy nodes.

The ACC also notifies the participants when any changes occur on their access level. To do so, the ACC uses the "status" field, the timestamp, and the timer units.

3.3.8. Mining Contract (MC). MC preserves the detailed "log" of each transaction carried out on the stored scientific data. It is designed to allow creating/validating/appending new blocks. MC lists the Eth.Add of the node which is considered as the "generator of the transaction." It includes the details of the transaction and a status field that demonstrates whether these details have been added to the blockchain or not. It also keeps the timestamp field which specifies the time of updating the status field.

3.3.9. The Proxy Reencryption Contract (PReC). To allow accessing scientific data by a "third-party-typeA" node, the proxy reencryption technique proposed by [28] is employed in the proposed design and is utilized by this contract. This contract will automatically be generated each time a new group of proxy nodes is determined.

The group of proxy nodes will be supplied by a shared private key and a master or a main public key. All nodes within the group of proxies have a unique private/public key pair.

Each node in the proxy group also has a public key that is known by others. Each node in the proxy set will pick randomly a blinding value "p" and encrypt it. The node then blinds that encrypted blinding value with the encrypted message. After that, portions of the blinded message will be decrypted on their own systems. The contribution of all proxy nodes will be maintained in the PReC. Each entry in the PReC includes (1) the Eth.Add of the proxy node, (2) the blinded plaintext message "mp," and (3) the encrypted pairs of p values.

3.3.10. Deposit-Box Contract (DBC). To allow accessing scientific data by a "third-party-typeB" node, the deposit-box schema is employed. A record will be encrypted first by the public key of the "third-party-typeB" node before being stored in this contract for a specific time stated by the timer field.

4. TimedChain Implementation

4.1. Initializing the TimedChain System: Part I—Adding a New Publisher Node. Before joining the blockchain system, publishers (of peer-reviewed journals/conferences) should accept the following: (1) the rules and policies of the proposed smart contracts; (2) the frequency of updating the blockchain network; (3) the proposed incentive mechanism; (4) the procedures of generating new blocks. Figure 3 shows the steps of adding a new publisher. Each publisher has a public identifier (ID) or an identification string that is unique to the official publisher. Each publisher first has to set up the blockchain to receive the Ethereum address (Eth.Add) and configure the software components. Note that the Eth.Add of a node is considered as a public key of that node. The process of adding a publisher node starts by sending the ID, the Eth.Add, and the "publisher" role to the NC. All transactions are timed to ensure the system performance. NC ensures that the received Eth.Add is unregistered previously. NC forwards the received request to the PC whose voters authenticate the received request and validate that it is commitment to a legal publisher (publisher). Upon validating and accepting the request, the NC adds the Eth.Add of the publisher to its local memory. The PC also adds the Eth.Add of the node and its ID. The PC generates a new RHC for the new node. The RHC.Add will be sent to the publisher node for later reference.

4.2. Initializing the TimedChain System: Part II—Evaluating the Value of a Publisher Node. This stage is performed by the publisher nodes listed in the PC. The process of reviewing articles is a key part in the publication process, ensuring that a publisher preserves high quality standards for its published papers. Sci-DEM, which is a python-based tool installed in each publisher node, evaluates the node's value by determining whether a stored record serves the purpose for which it was intended or not. This work evaluates the value of a node, δ_i, by measuring the quality and the quantity of the "review records" maintained in its database as

$$\delta_i = \sum_{\mathrm{RR}=1}^{m} Q_{\mathrm{RR}}. \tag{1}$$

This paper proposes a general measurable standard for a record's quality. The quality of a record is evaluated by measuring the record's Consistency, Correctness, and Completeness (3C's), in addition to the nonredundancy attribute.

(i) Completeness (CM): A complete record includes all fields, items, and elements that all joining publishers have agreed on and accepted during the initialization stage. For instance, a review report is considered complete if the reviewer fills in the required fields.

(ii) Correctness (CR): The correctness of a scientific record is evaluated by measuring data accuracy.

(iii) Consistency (CN): A consistent record means that all data the record includes are reliable.

(iv) Nonredundancy (NR): It demonstrates that the data stored in a record is not repeated by more than one publisher node. For instance, review reports that have the same comments are considered as redundant reports.

To measure the NR attribute, Sci-DEM will check the similarity of data stored in a record. The nonredundancy of record "I" is equal to 1 (NRI = 1) when all record's data are not repeated (i.e., unique) by other publishers; otherwise, the value of the nonredundancy attribute will be divided between those publishers who share the same data.

To measure the 3C's attributes, Sci-DEM will categorize and classify each field, item, and element in the stored record as follows: n_1: correct; n_2: incorrect; n_3: missing; n_4: extra; n_5: conflict. Therefore,

$$\begin{aligned} \mathrm{CM}_{\mathrm{RR}} &= \frac{\sum n_1 + n_2 + n_5}{\sum n_1 + n_2 + n_3 + n_5}, \\ \mathrm{CR}_{\mathrm{RR}} &= \frac{\sum n_1}{\sum n_1 + n_2 + n_4 + n_5}, \\ \mathrm{CN}_{\mathrm{RR}} &= 1 - \frac{\sum n_5}{\sum n_1 + n_2 + n_4 + n_5}. \end{aligned} \tag{2}$$

Accordingly, the publisher's value is evaluated by

$$\delta_i = \sum_{\mathrm{RR}=1}^{m} Q_{\mathrm{RR}} = \sum_{\mathrm{RR}=1}^{m} \mathrm{CM}_{\mathrm{RR}}\mathrm{CN}_{\mathrm{RR}}\mathrm{CR}_{\mathrm{RR}}\mathrm{NR}_{\mathrm{RR}}. \tag{3}$$

At the end of this stage, all publishers, as shown in Figure 4, will add/update their values listed in the PC. A publisher node sends a request to the NC to add/update its value. NC will first validate if the request is received from a registered node. The NC then forwards the request to the PC to "add/update" the value of the node. After that, the average value of nodes within the system, voters' nodes, and a block creator node will automatically be updated in the PC.

It is worth noting that the node is labeled as a "voter" node if it has a value equal to or greater than the average value of the network, while the node with lower value in the network will be labeled as a "block creator" node, which will be assigned the task of creating/verifying/appending the next new block.

4.3. Adding a New Author Node. This stage is shown in Figure 5. It begins when the author via a web interface sends a "joining request" to the publisher node (publisher) which in turn forwards the required configuration steps to set up an author node and create an Eth.Add.

The Eth.Client of the publisher node sends the information of the new author node to the blockchain. For validation, it sends the node ID, Eth.Add, and the "author" role to the NC. NC ensures that the received Eth.Add is unregistered previously. NC forwards the received request to the PC whose voters authenticate that request and validate that it is commitment to a legal author who is not registered previously. Upon validating and accepting the request, the NC adds the Eth.Add of the author to its local memory.

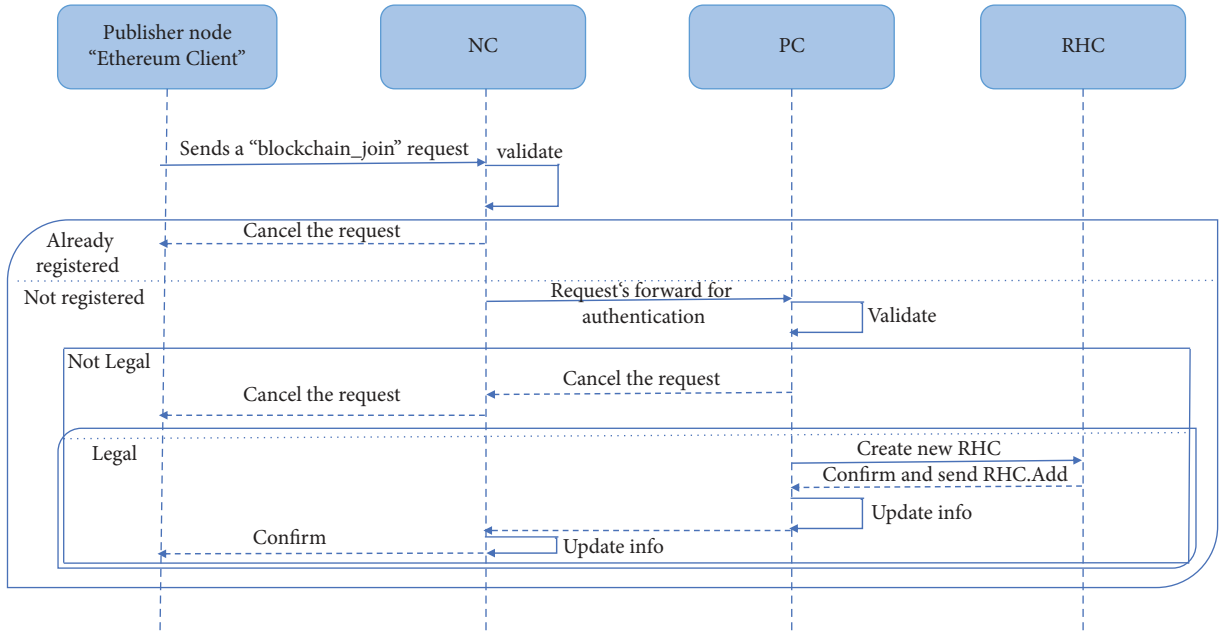

Figure 3: Adding a new publisher node.

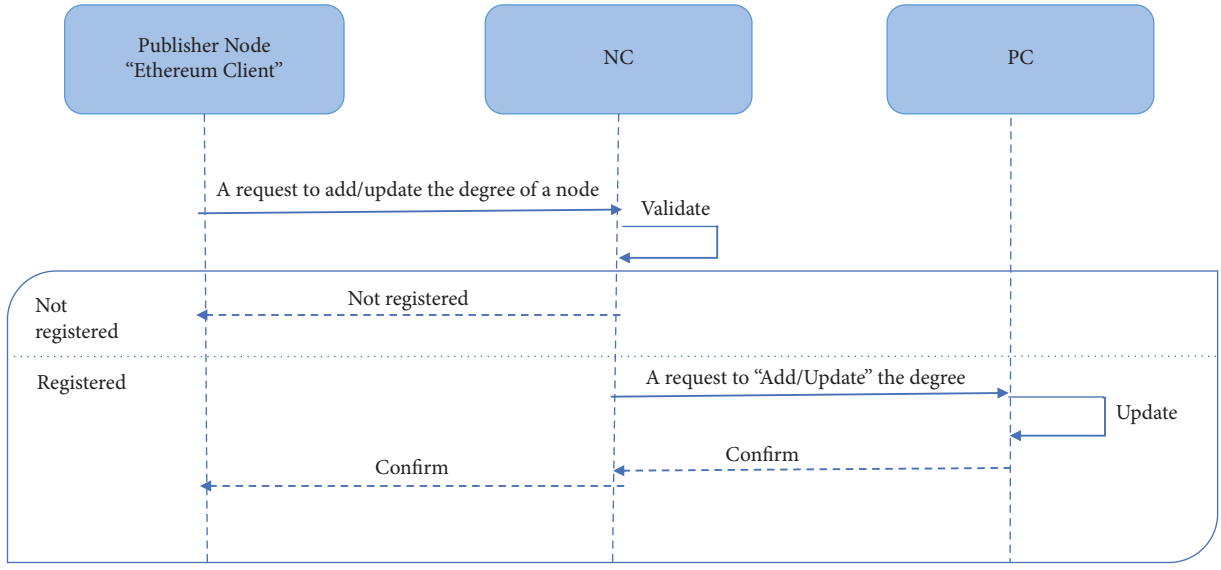

Figure 4: Adding/updating the value of a node.

Furthermore, the AC adds the Eth.Add of the node and its ID. The AC generates a new RHC for the new node. The RHC.Add will be sent to the publisher node. The new author's account information will be sent to the author node from the publisher node that forms the request.

4.4. Author Registration. Once the author has joined the blockchain system, the author registration stage is started (see Figure 6). In this stage, a “registration request” will be sent from the web interface of the author node to the web interface of the publisher node which in turn will forward the Eth.Add of the author and its “author” role to the NC for verification. The NC verifies the “author” node role through the AC, which will verify that the registration process is accomplished for the new author node, which joined the system. Otherwise, an author node has to be added first (see stage C).

After that, the author information including its ID and Eth.Add is transmitted to the RCH of the publisher node, which will check the author originality. The RHC indicates whether the author is new in the peer-review system or it has been registered previously. Upon confirmation, the RHC requests the generation of new stewardship with the author who can decide to either accept or reject that request. As a result, the RHC generates a new entry that includes the Eth.Add of the author node, its ID, and “newly established-pending” status, and updates the “timestamp” data field accordingly.

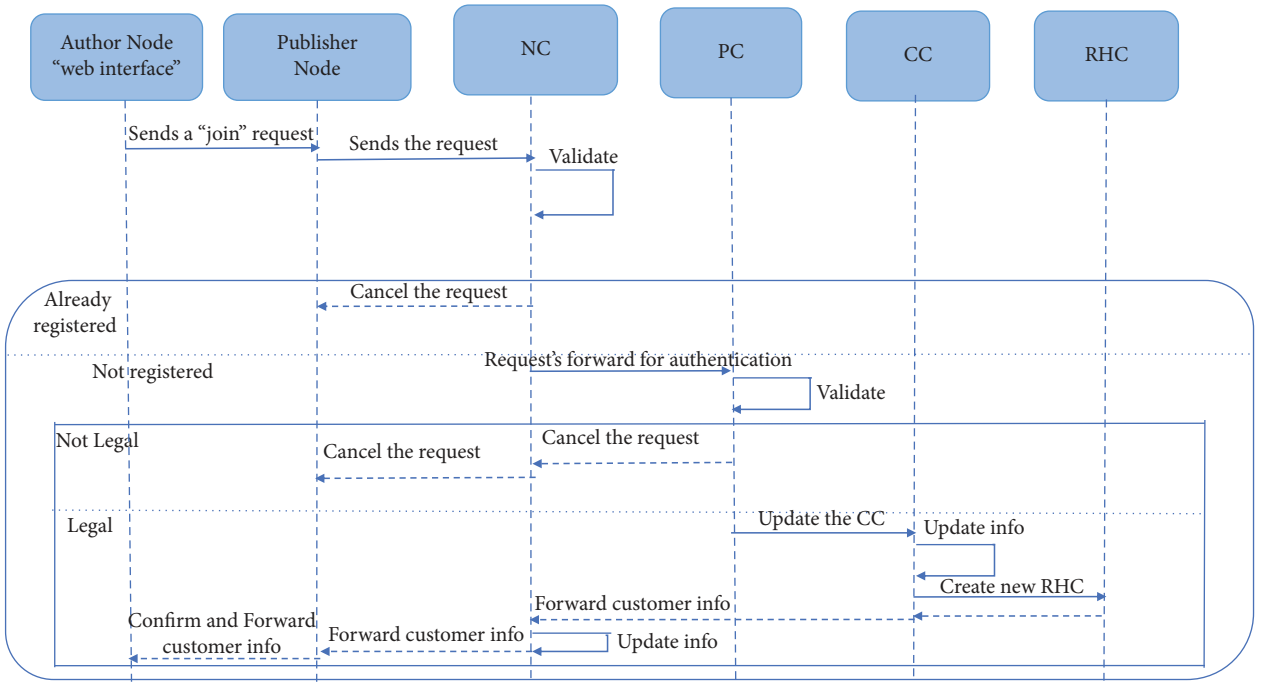

Figure 5: Adding a new author node.

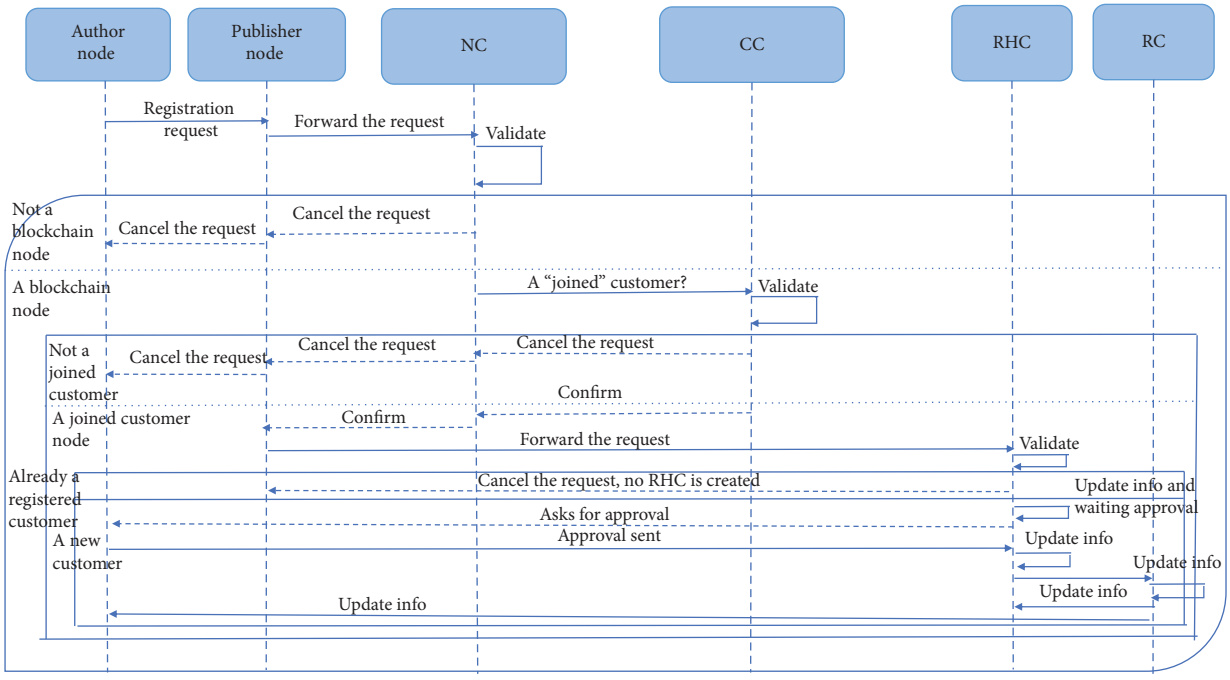

Figure 6: Author registration: establishing stewardship between the publisher (peer-reviewed system) and the author nodes (the author).

Similarly, at the author node, the RHC creates a new entry that includes the Eth.Add of the publisher node, its ID, and "newly established-pending" status, and updates the "timestamp" data field.

Upon the author acceptance, the status field of the RCH is updated as "approved" along with the timestamp field, for both the publisher and the author nodes. Otherwise, the process will be canceled. After that, the RHC of the publisher node creates a new RC for the new stewardship. The RC fills in the author's related information and sends its address to the RHC to update the "RC.Add" data field, for the publisher and the author nodes.

4.5. Manuscript Submission (Record). The process of submitting a new manuscript, shown in Figure 7, begins after establishing stewardship (i.e., author registration) between the publisher (peer-reviewed system) and the author nodes

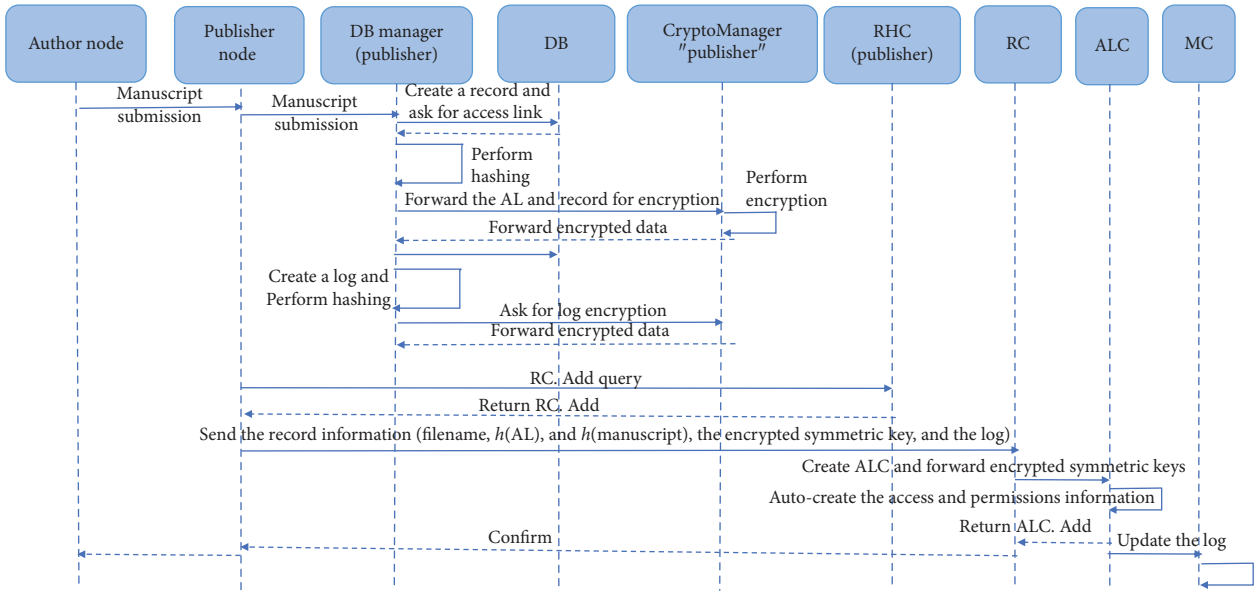

Figure 7: Submitting a new manuscript.

(the author), thus having a shared RC. The web interface of the author node sends a request of submitting a new manuscript along with the manuscript to the peer-reviewed system (publisher).

Once the publisher node receives the author node's request, it will start the process of submitting a new manuscript by generating the editor(s)/reviewers' reports and performing internal encryption. The new manuscript and the review reports will be forwarded to the DB manager of the publisher node to generate access links of free locations in the existing database. The DB manager of the publisher takes a further step by hashing both the received manuscript and the generated access links, which are sent to the CryptoManager for encryption.

The CryptoManager in turn encrypts the received data in two steps. At first, a symmetric key is generated, which is used to encrypt the access links, the manuscript, and the generated review reports. After that, the generated symmetric key will be encrypted using the public keys of the publisher (i.e., editor), author (i.e., author), and set of proxies. At the end, the encrypted records are forwarded and stored in the DB manager.

Further, the DB manager of the publisher node tracks the process of submission the new manuscript by creating a "log" file which tracks the access to the manuscript and other scientific data. The log file will be hashed and stored in the blockchain to afford a full view of all events that happened to all stored records. Then, the CryptoManager encrypts the log file using the public key of the publisher node in order to allow verifying the created block later. Ultimately, this will enhance the security and the integrity of the data.

The Eth.Client of the publisher node requests the associated RC.Add from the RHC using the author's ID (the ID of the author). The Eth.Client of the publisher node then sends the record information (filename of the record, hash values of the links, hash value of the manuscript, the encrypted symmetric key, and the log) to the RC. The RC stores the received information. The RC then creates a new ALC for the record(s) and forwards the encrypted keys E_{pk} *(SMK)* to the ALC. The ALC auto-creates the access and permissions information for the record and then sends its address to the RC for its reference.

On the other hand, the MC adds a new entry that includes the encrypted log and the Eth.Add of the publisher node (i.e., source of the transaction). Accordingly, the MC status is defined as "new log," which indicates that the new entry has not been added to the blockchain yet, and updates its timestamp field to specify the time of the last modification on the status data field. At the end, the web interface of the publisher node sends the encrypted access link of the submitted manuscript to the web interface of the author node over HTTPS which in turn forwards it to the CryptoManager to store. The received link can be used by the author node later for viewing the submitted manuscript.

4.6. Manuscript Handling. The request of reading/updating a stored record will be sent from the web interface of the publisher node to its backend library, which in turn will parse and translate the request and send it to its Eth.Client. The Eth.Client sends the author's ID to the RHC of the publisher node to fetch the associated RC.Add. The Eth.-Client of the publisher node then sends to the RC both the record's filename and the publisher's Eth.Add. The RC forwards the address to the ALC to check and verify its access permission (i.e., "owner" access level).

Upon verification, the ALC forwards the publisher's encrypted symmetric key E_{pk} *(SMK)* to the RC. The RC in turn forwards the received key to the Eth.Client of the publisher node. The Eth.Client sends via the backend library the received encrypted symmetric key to its CryptoManager. The CryptoManager uses its private key to decrypt the

received symmetric key, which will be used to decrypt the access link stored in the publisher's DB manager. The decrypted access link will be used by the DB manager of the publisher's node to retrieve the encrypted record from the database for reading/updating activities.

It is worth noting that the hash value of a record is changed as the record status is updated. Accordingly, the DB manager of the publisher node generates the new hash value of the updated record. Then, the DB manager requests the RHC to send the RC.Add that is associated with the author's ID. At the end, the RC is updated with the new hash value.

In addition, the DB manager of the publisher node creates a log to track the process of updating the records. The log will be hashed and forwarded to the CryptoManager, which will be encrypted and forwarded to the MC. The MC in turn adds a new entry that includes the encrypted log and the Eth.Add of the publisher node (i.e., source of the transaction). Accordingly, the MC status is defined as "new log," which indicates that the new entry has not been added to the blockchain yet, and updates its timestamp field to specify the time of the last modification on the status data field. It is worth noting that when a record is updated, the publisher node notifies the PC to update the record's associated value. Then, the PC will inform the Sci-DEM to update the node's value according to the new updates of the record. At the end, the Sci-DEM calculates the new value, which will be sent to the PC for updating.

4.7. Reading the Submitted Manuscript from an Author Node. Figure 8 specifies the steps of reading a manuscript. The request of reading a submitted manuscript will be sent from the web interface of the author node to its backend library, which in turn parses and translates the request and sends it to its Eth.Client. The Eth.Client sends the publisher's ID to the RHC to fetch the associated RC.Add. The Eth.Client of the author node then sends to the RC both the record's filename and the author's Eth.Add. The RC forwards the address to the ALC to check and verify its access permission.

Upon verification, the ALC forwards the author's encrypted symmetric key E_{pk} *(SMK)* to the RC. The RC in turn forwards the received key to the Eth.Client of the author node. The Eth.Client sends via the backend library the received encrypted symmetric key to its CryptoManager, which uses its private key to decrypt the received symmetric key. The CryptoManager then uses the decrypted key to decrypt the access link stored in its DB manager. The DB manager of the author's node retrieves the encrypted record from the database by using the related access link. In addition, the DB manager creates a log to track the process of updating the records. The log will be hashed and forwarded to the CryptoManager, which will be encrypted and forwarded to the MC. The MC in turn adds a new entry that includes the encrypted log and the Eth.Add of the publisher node (i.e., source of the transaction). Accordingly, the MC status is defined as "new log," which indicates that the new entry has not been added to the blockchain yet, and updates its timestamp field to specify the time of the last modification on the status data field.

4.8. External Reviewer Invitation. All manuscripts submitted to publisher are reviewed by at least two researchers (i.e., experts) in the field. A reviewer will evaluate the scientific quality of an assigned manuscript, for which he/she will provide a recommendation for the external editor; for example, the manuscript can be accepted, requires revisions, or should be rejected. In the proposed design, a third-part-typeA node represents a reviewer. Assume that a reviewer (third-party-typeA) is invited to review a manuscript submitted by an author. Figure 9 shows the steps of sending an invitation to review a manuscript.

To send the review invitation, the Eth.Client of the publisher node sends the ID of the author, i.e., author, to its RHC to fetch the associated RC.Add. The Eth.Client of the publisher node then sends the filename of the record (i.e., manuscript title) to the RC to fetch the "ALC.Add" and "MC.Add." The Eth.Client of the publisher node then sends the Eth.Add of the reviewer and the "access level" request to the ALC. The ALC then forwards the request to the NC which in turn forwards it to the TPC for verification.

Upon receiving the verification, the ALC creates a new entry with the reviewer's Eth.Add, the access level, and the "request new level" status. The ALC updates its timestamp field to specify the time of the last status update. The ALC requests changing the access level from the reviewer and updates both its status field to "invitation_sent" and timestamp field to specify the time of the last status update. Upon accepting the request by the reviewer, the access level for the applicable file will be updated with the "review_invitation_accepted_review_waiting" status. Once the request has been approved, the status field will be updated and a notification will be sent to the reviewer indicating that a new manuscript has been assigned for review.

The DB manager of the publisher adds a new entry to the MC indicating the "review invitation." The MC adds a new entry that includes the encrypted log and the Eth.Add of the publisher node (i.e., source of the transaction). The MC changes its status to "new log" to demonstrate that the new entry has not been added yet to the blockchain and updates its timestamp field to specify the time of the last status update.

4.9. Review Report Submission. The process of reviewing a manuscript by external experts, as shown in Figure 10, employs the proxy reencryption schema to enhance the security of the peer-reviewed systems and to improve the system's accessibility. This process includes two subprocesses: (1) retrieving the submitted manuscript; (2) retrieving and submitting a review report.

To retrieve the manuscript and submit a review, a reviewer node (web interface) generates a request to submit a review report for the assigned manuscript. For authorization, the CryptoManager of the reviewer node, first, signs that created request with the reviewer's private key. Then, the CryptoManager of the reviewer encrypts the signed request with the public key of the publisher (publisher). The web interface of the reviewer node over HTTPs sends the signed request to the publisher web interface. The request

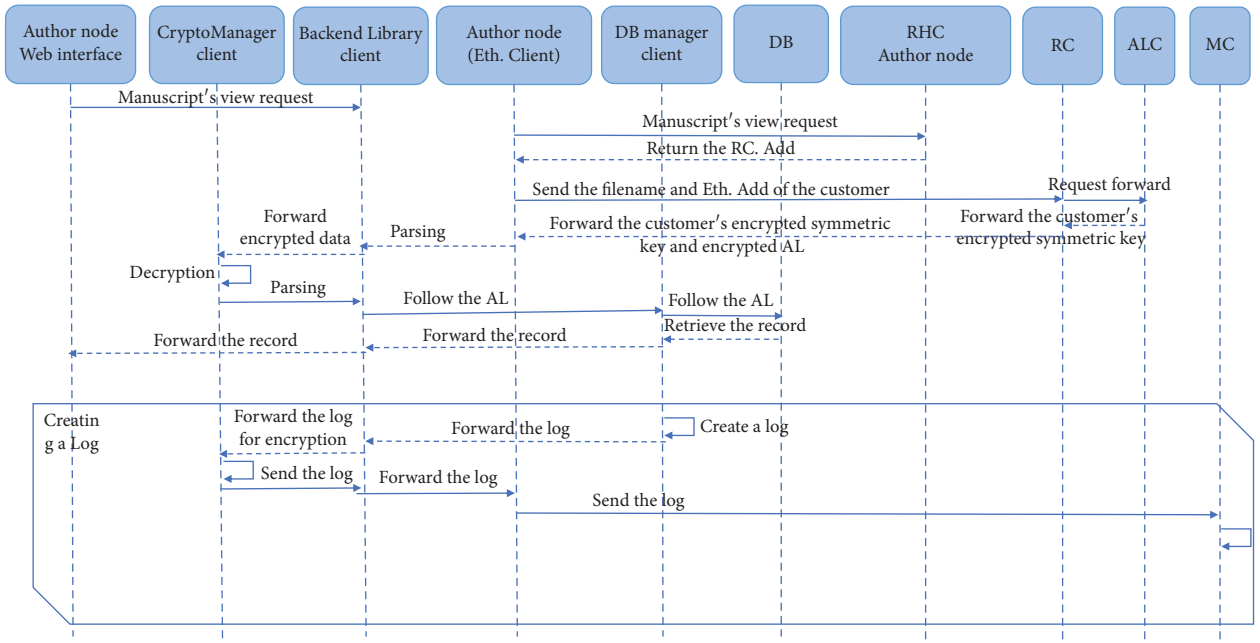

Figure 8: Reading a manuscript.

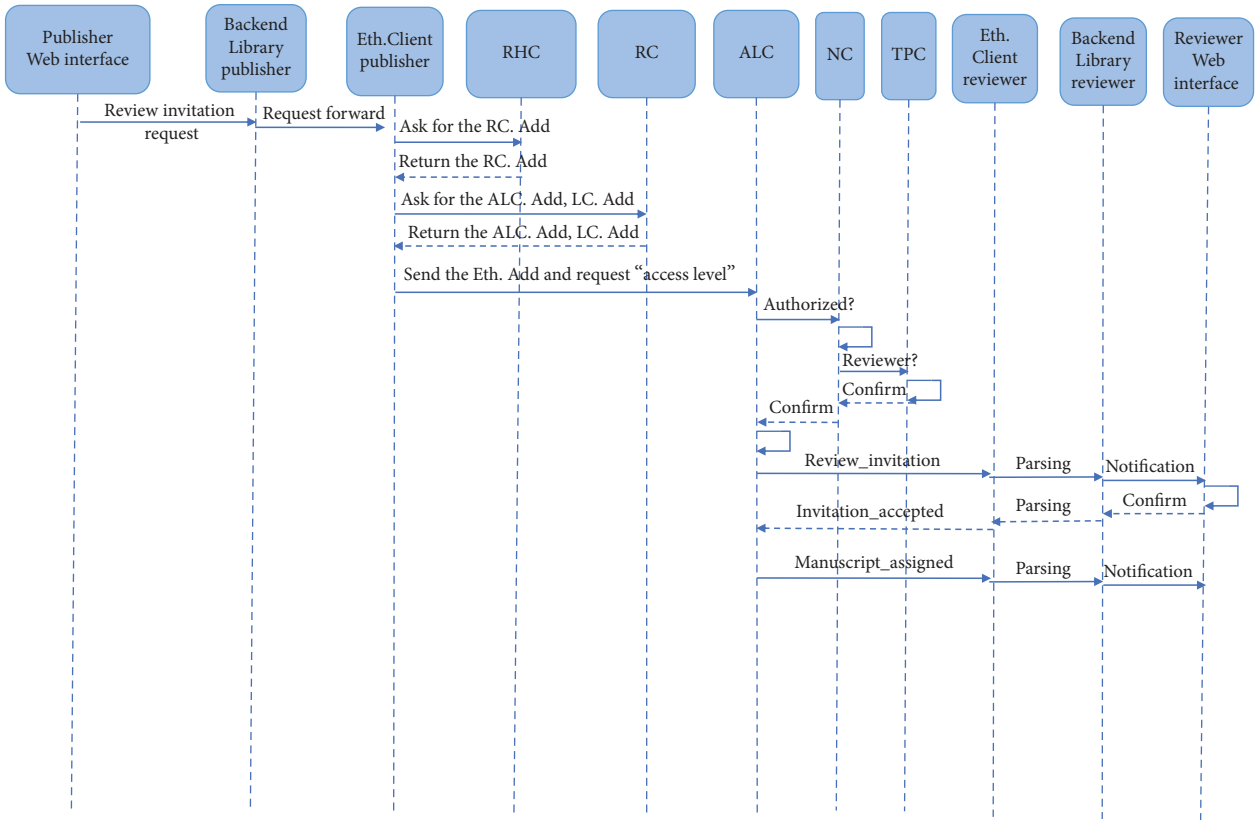

Figure 9: Invitation to review a manuscript.

will be received by the web interface of the publisher node which in turn via its backend library will be forward it to the publisher's CryptoManager. The CryptoManager decrypts the request with the private key of the publisher node and then decrypts it with the public key of a reviewer node to assure that the reviewer is the one that it claims to be.

The Eth.Client of the publisher node sends the ID of the author, i.e., author, to its RHC to fetch the associated

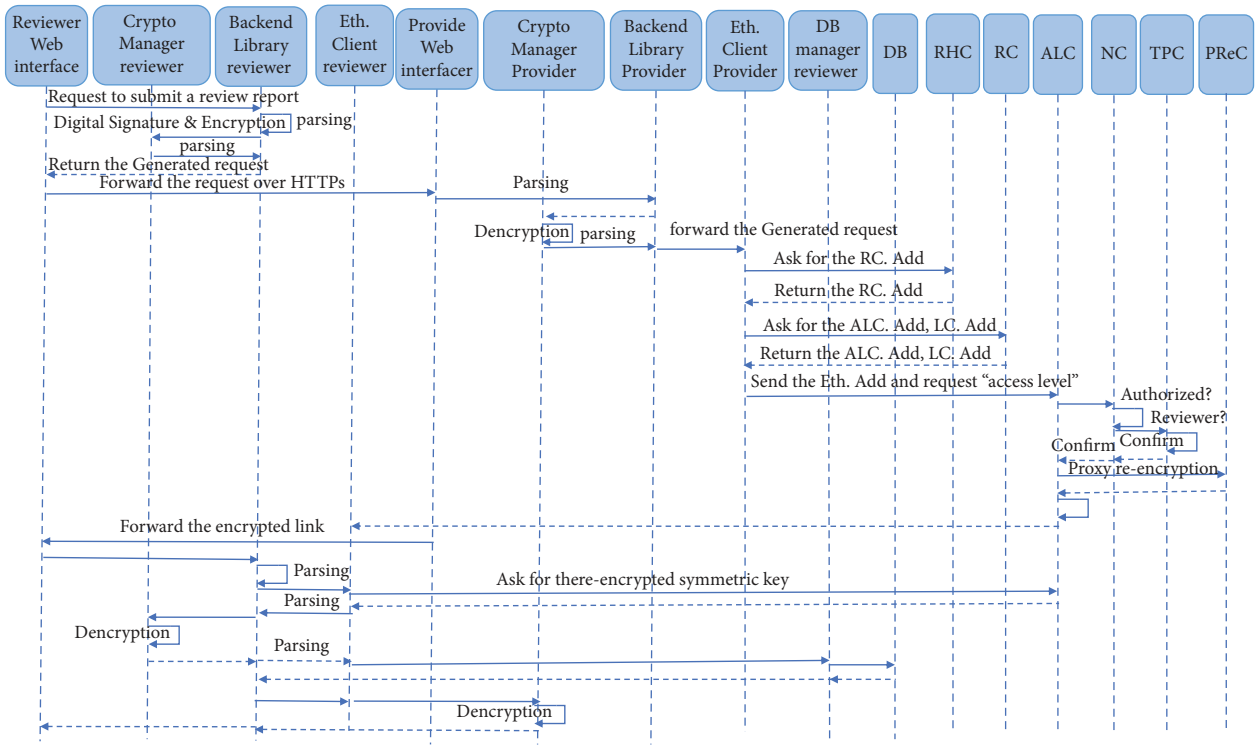

Figure 10: Manuscript review.

RC.Add. The Eth.Client of the publisher node then sends the filename of the record (i.e., manuscript title) to the RC to fetch the "ALC.Add" and "MC.Add." The Eth.Client of the publisher node then sends the Eth.Add of the reviewer and the "access level" request to the ALC. The ALC then forwards the request to the NC which in turn forwards it to the TPC for verification. The Eth.Add of a reviewer and the master encrypted symmetric key will be sent to the proxy nodes by the ALC to perform the proxy reencryption schema. The reencrypted symmetric key will be kept in the ALC which in turn sends its ALC.Add to the reviewer node.

Over HTTPS, the web interface of the publisher node sends the encrypted query link of the manuscript and the review report to a reviewer node that can decrypt the links and fetch the records. A reviewer node has the ALC.Add that will be used to fetch the stored "reencrypted symmetric key." The CryptoManager of the reviewer node decrypts the "reencrypted symmetric key" using its private key and then decrypts the query links with that symmetric key. The DB manager of the reviewer node follows the query links to retrieve the submitted manuscript and the review report. The manuscript and the report will appear to the reviewer after decrypting it using the symmetric key.

When the review report is submitted, a notification will be sent to the publisher node. Accordingly, the DB manager of the publisher node generates the new hash value of the updated record. Then, the DB manager requests the RHC to send the RC.Add that is associated with the author's ID. At the end, the RC is updated with the new hash value. Moreover, the DB manager creates a log to track the process of submitting a review report. The log will be hashed and forwarded to the CryptoManager, which will be encrypted and forwarded to the MC. The MC in turn adds a new entry that includes the encrypted log and the Eth.Add of the publisher node (i.e., source of the transaction). Accordingly, the MC status is defined as "new log," which indicates that the new entry has not been added to the blockchain yet, and updates its timestamp field to specify the time of the last modification on the status data field. Then, the PC will inform the Sci-DEM to update the node's value according to the new updates of the record. At the end, the Sci-DEM calculates the new value, which will be sent to the PC for updating.

4.10. Third-Party-TypeB Node (Researcher) Reads a Record from a Publisher Node. The process of reading a record by a third-party-typeB node utilizes the time-based deposit-box schema to increase both the security of the peer-reviewed system and its accessibility.

A third-party-typeB node generates a request to read a record. For authorization, the CryptoManager of the node, first, signs that created request with the node's private key. Then, the CryptoManager encrypts the signed request with the public key of the publisher (publisher). The web interface of the node over HTTPs sends the signed request to the publisher web interface. The request will be received by the web interface of the publisher node which in turn via its backend library will forward it to the publisher's CryptoManager. The CryptoManager decrypts the request with the private key of the publisher node and then decrypts it with the public key of a third-party-typeB node to assure that it is the one that it claims to be.

The Eth.Client of the publisher node sends the ID of the author, i.e., author, to its RHC to fetch the associated RC.Add.

The Eth.Client of the publisher node then sends the filename of the record (i.e., manuscript title) to the RC to fetch the "ALC.Add" and "MC.Add." The Eth.Client of the publisher node then sends the Eth.Add of the node and the "access level" request to the ALC. The ALC then forwards the request to the NC which in turn forwards it to the TPC for verification.

Once receiving the verification, the ALC creates a new entry with the Eth.Add of a third-party-typeB node, the access level, the "request new level" status, and the timestamp of the last status update. The ALC requests changing the access level from the file owner and updates both its status field to "waiting approval" and timestamp field to specify the time of the last status update. Upon the owner acceptance of the request, the ALC will update the status of the access level as "temp-read" and the status of the applicable file as "approved," and it will update the timestamp. When the request is approved, the ALC notifies a third-party-typeB node that it is being assigned a new access level. The ALC also notifies the RC to create a new entry in the DPC associated with the Eth.Add of the third-party-typeB node. The RC then sends the Eth.Add of a third-party-typeB node and the filename to the publisher node. Then, the DB manager retrieves the record and forwards the record with the received public key to the CryptoManager for encryption.

The DBC will be updated by storing the encrypted file for a certain time specified by the time field. Over HTTPS, a publisher sends the DPC.Add to a third-party-typeB node to access the requested record by decrypting it using its private key. The DB manager of a third-party-typeB node will update the MC entry with an encrypted log indicating the process of reading the record. The MC adds a new entry that includes the encrypted log and the Eth.Add of the source of the transaction. The MC updates its status field to be "new log" to demonstrate that this new entry has not been added yet to the blockchain and updates its timestamp field to specify the time of the last modification on the status data field.

4.11. Creating, Verifying, and Appending a New Block. To perform the task of creating a new block, there is a need to select a publisher node that has the lowest value among all publishers within the blockchain network. This publisher will be notified by the PC to perform the computational task of creating the new block. Publishers who maintain more valuable scientific data are less likely to be selected (see Figure 11).

To illustrate the procedures of creating/verifying/and appending a new block, assume that there is a blockchain network of 3 publishers and 10 authors. Author "c.A" reads a record; Author "c.B" updates a record; Publisher "P.B" creates 3 new records. Logs related to all Create/Read/Update/Delete (CRUD) operations will be saved in the MC. Assume that publisher "P.A" is selected to create a new block.

Publisher "P.A" first sends a request "of creating a new block" to the MC. The MC in turn asks the NC if publisher "P.A" is an authorized publisher on the system. The NC forwards the request to the PC to ensure that publisher "P.A" is the one who is selected to perform the task. After receiving the verification, the MC sends to publisher "P.A" all the encrypted logs listed with status "new log". Publisher "A" then (1) creates a new block including all received logs, (2) broadcasts the new block, and (3) calls for verification.

Logs verification will be performed by all involved nodes through (1) decrypting the log with its private key, (2) hashing the log after decryption, (3) comparing the hash result with the one listed in its DB manager, and (4) sending a signed proof to publisher "A."

Publisher "A" then, after receiving all the signed proofs, notifies the PC to update its value. The PC adds the rewarded incentive value "c" to the current value of the publisher "A." The publisher who will be picked to generate the new block will be rewarded an incentive "c" upon successfully verifying the block by other involved nodes. The distribution of publishers' values and the number of nodes within the blockchain network affect the value of the rewarded incentive "c." Thus, "c" is defined as a fraction of the average values in the network.

Publisher "A" broadcasts to all publishers to append the new block. After appending the new block, the MC automatically updates the status field for all logs which are added to the chain as "appended" and fills in the timestamp field.

5. TimedChain Evaluation

5.1. Experimental Setting. To evaluate the proposed system, a computer system with 2.6 GHz Intel Core i7 processor, 16 GB 2400 MHz DDR4 memory, and macOS High Sierra operation system is used. For implementation, the open source, Ethereum platform, is utilized. Solidity is used for scripting the functionalities of the proposed smart contracts. Truffles are used to deploy the proposed smart contracts as it has no restrictions regarding the capacity and the size of the stored data. The web3.js library is employed to enable interacting with an Ethereum-based node using HTTPS. For testing the CRUD services, Apache JMeter is applied as a performance measuring mean and a functional tool. The experiments' parameters and their values are shown in Table 1.

As shown in the table, three parameters are used to conduct the experiments, namely, the number of CRUD queries, the number of registered nodes, and the number of stored records within a node. In the conducted experiments, the query time, the communication overhead, and the average response time are measured. For testing and to study the relation between the results and the parameters, only one parameter was modified each time.

5.2. Results and Discussion. The results of all experiments indicate that the query time, the communication overhead, and the average response time increase as the total stored

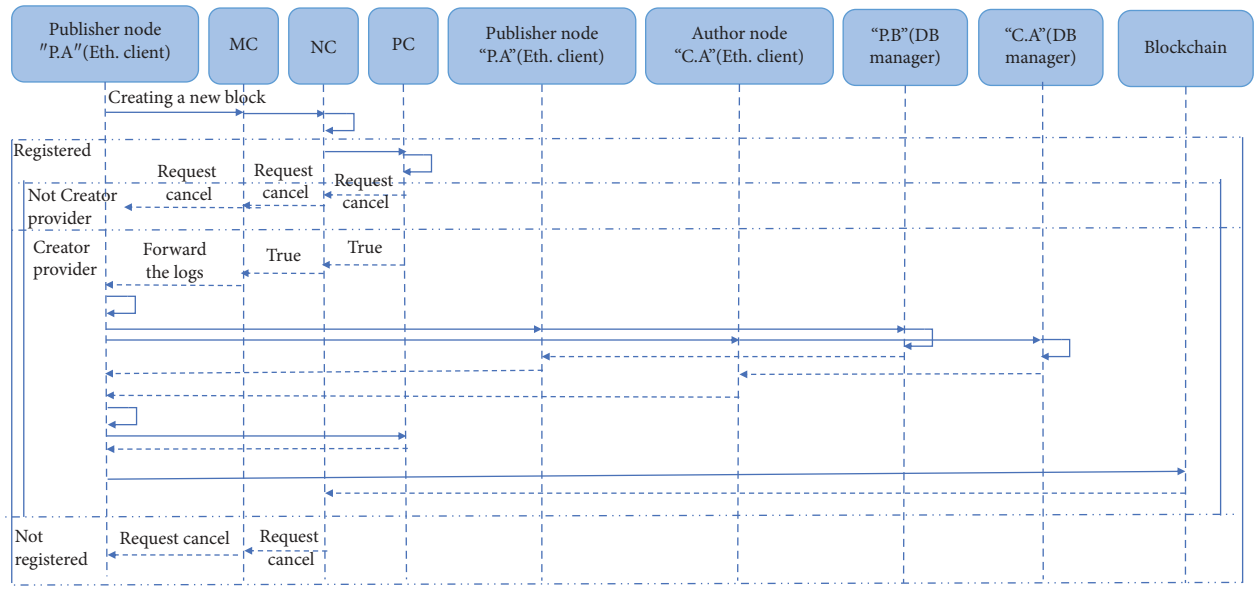

Figure 11: Creating/verifying/appending new block.

Table 1: Experiments' parameters.

Description	Values
Number of nodes joining the blockchain	10,000–100,000
Number of records stored in a node	10,000–100,000
Number of CRUD queries	10,000–100,000

records, the total submitted CRUD queries, and total number of nodes were increased, as shown in Figures 12–14.

The results show that submitting more CRUD queries causes storing more records which means spending more time to complete these queries. Moreover, increasing the number of nodes that join the system means using the system by more participants which in turn increases the number of submitted queries and increase the time for queries to be completed, hence increasing the communications among participants' nodes.

Nevertheless, as shown in Figure 15, the throughput remains steady and stable even with raising the number of the submitted queries, the number of records stored in the nodes, or the number of nodes within the TimedChain network. This stability in the throughput demonstrates the power of the system in managing and processing a large size of data with high frequency at low latency.

The main reason behind these results is the balance between the "on-chain" (i.e., actions performed through the blockchain network) and the "off-chain" (i.e., actions performed outside the blockchain network) operations of the proposed system. The actions classified as "off-blockchain" actions include the following: (1) nodes' values evaluation when initializing the blockchain; (2) generating a record's access link when creating a record; (3) record's hashing/encrypting/decrypting; (4) procedures of database storage and retrieval. On the other hand, the "on-blockchain" actions involve the following: (1) storing and acquiring data in and from smart contracts; (2) internal transactions between the proposed smart contracts; (3) generating and creating

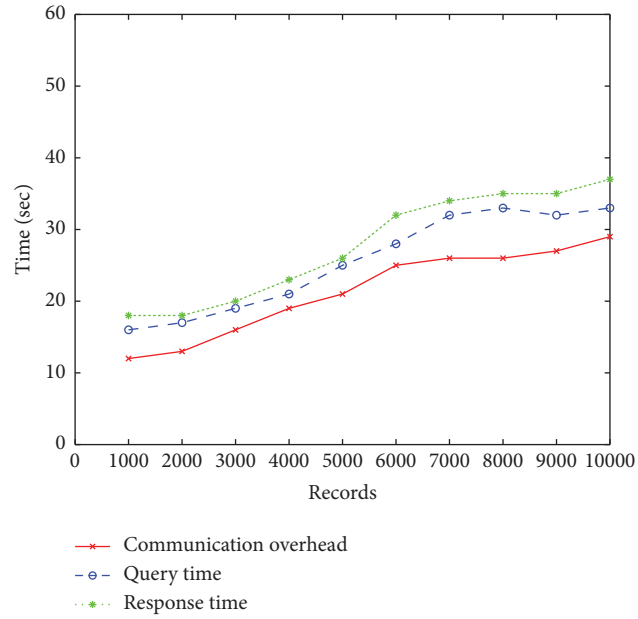

Figure 12: Results of system performance (changing the number of stored records).

new contracts via other contracts. The utilization of "on/off-chain" actions plays a key role in improving the proposed TimedChain system and maintains the system performance.

Moreover, there is a critical impact of adopting the PoA consensus algorithm on the system performance. In comparison with the PoS or the PoW, the PoA has the ability to manage more transactions per second as it decreases the time-consuming computations and maximizes the system performance. The PoA entails lower latency in accepting transactions and stable time intervals for issuing blocks. Regarding the proposed system, generating and validating a new block require two rounds.

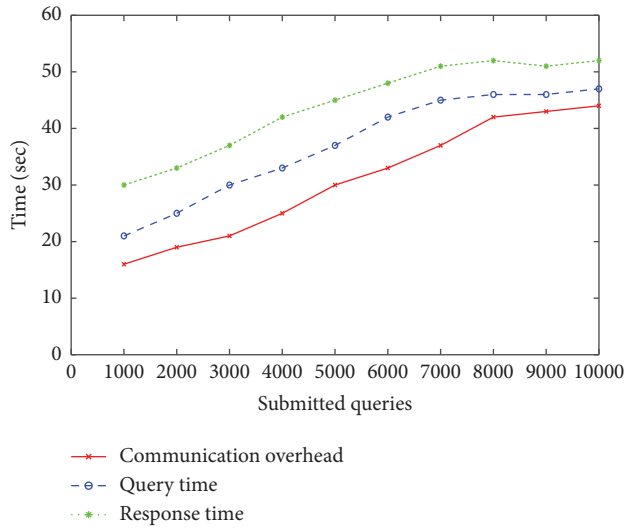

Figure 13: Results of system performance (changing the number of submitted queries).

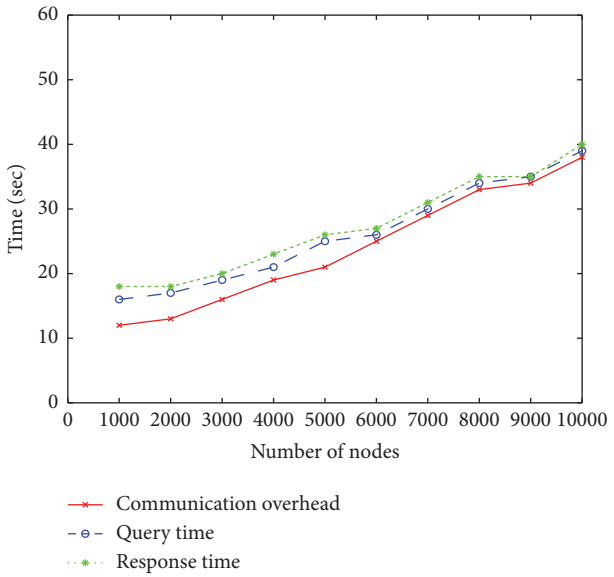

Figure 14: Results of system performance (changing the number of nodes).

The "block's creator" node (i.e., the publisher node that has the lowest value among other publishers within the system) forwards the new created block to involved nodes only (i.e., an involved node is a publisher node that is the source of the performed transaction). It is not required to dispatch the block to all network's participants, thereby decreasing the communication overhead. In round 2, the created block will be verified by those involved nodes. Block verification demonstrates approving the list logs. Appending the new block will be accomplished upon receiving the verifications from all involved nodes. It is clear that the adoption of the PoA minimizes the number of messages sent for generating and validating new blocks which has a notable impact on enhancing the performance of the system.

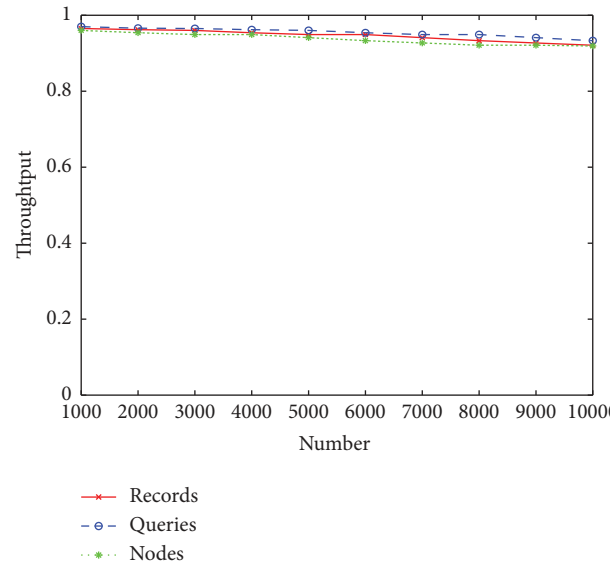

Figure 15: Throughput of the system.

Additionally, employing time-based smart contracts guarantees acceptable period of time for performing transactions and computations. Timers will be reset to zero and data will be destroyed when there is a loss of connectivity.

6. Conclusion

This paper proposes the TimedChain system, which is a blockchain-based peer-review system that facilitates the integration with existing databases owned by publishers. The TimedChain provides secure, interoperable, and efficient access by publishers (of peer-reviewed journals/conferences), authors (authors), and third parties (external editors/researchers/readers). According to the proposed design, maintaining the blockchain network by creating/verifying/appending new blocks is the role of publishers. Moreover, publishers securely control access to scientific data stored in the databases.

Privacy is ensured by exploiting time-based smart contracts for controlling, managing, and governing transactions. The proposed smart contracts monitor the computations and the queries carried out on the scientific data by implementing valid usage policies. Data integrity is achieved by applying the hashing techniques. Access control and security are ensured by employing advanced authentication and encryption techniques (i.e., public key encryption, symmetric key encryption, proxy reencryption, and deposit-box). Auditability, interoperability, and accessibility are met by utilizing and employing comprehensive logs.

For mining, this research introduces a new incentive mechanism that is combined with the PoA. It estimates the value of publishers respecting their efforts at managing publications and creating new blocks. A publisher who has the lowest value among all publishers will be picked to create the next new block. An incentive will be rewarded to the "block's creator" node and added to the node's value to

minimize its possibility of recreating the next block, hence reaching fairness status and achieving system's sustainability.

Data Availability

The data supporting this study are available within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors thank Palestine Technical University-Kadoorie (PTUK) and Al Istiqlal University for their support.

References

[1] ScholarOne, 2020, https://clarivate.com/webofsciencegroup/solutions/scholarone/.
[2] Elsevier Submission System, 2020, https://www.elsevier.com/editors/submission-systems.
[3] Editorial Management, 2020, https://www.ariessys.com/.
[4] Bepress, 2020, https://www.bepress.com/.
[5] J. Willinsky, "Open journal systems: an example of open source software for journal management and publishing," *Library Hi Tech*, vol. 23, pp. 504–519, 2005.
[6] EJManager, 2020, https://www.ejmanager.com/?page=ejmfeatures.
[7] S. Kim, H. Choi, N. Kim, E. Chung, and J. Y. Lee, "Comparative analysis of manuscript management systems for scholarly publishing," *Science Editing*, vol. 5, pp. 124–134, 2018.
[8] T. K. Mackey, N. Shah, K. Miyachi, J. Short, and K. Clauson, "A framework proposal for blockchain-based scientific publishing using shared governance," *Frontiers in Blockchain*, vol. 2, p. 19, 2019.
[9] J. Yoo, H. Lee, T. Lee, and M. Shin, *Pluto. Breaking Down the Barriers in Academia*, 2018, https://assets.pluto.network/Pluto_white_paper_v04_180108_2130_BSH.pdf.
[10] ARTiFACTS, 2020, https://www.prnewswire.com/news-releases/artifacts-launches-first-ever-blockchain-based-platform-for-scientific-and-scholarly-research-300615989.html.
[11] Sciencematters, 2020, https://www.sciencematters.io/.
[12] Orvium. Architecture & Technology Specification, https://orvium.io/.
[13] S. Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, https://bitcoin.org/bitcoin.pdf.
[14] E.-Y. Daraghmi, Y.-A. Daraghmi, and S.-M. Yuan, "MedChain: a design of blockchain-based system for medical records access and permissions management," *IEEE Access*, vol. 7, pp. 164595–164613, 2019.
[15] E.-Y. Daraghmi and Y.-A. Daraghmi, "UniChain: a design of blockchain-based system for electronic academic records access and permissions management," *Applied Sciences*, vol. 9, p. 4966, 2019.
[16] N. Szabo, "The idea of smart contracts," vol. 6, 1997 Nick Szabo's Paper Concise Tutor.
[17] R. Modi, *Solidity Programming Essentials: A Beginner's Guide to Build Smart Contracts for Ethereum and Blockchain*, Packt Publishing Ltd, Birmingham, UK, 2018, ISBN 978-1-78883-138-3.
[18] C. Cachin, *Architecture of the Hyperledger Blockchain Fabric*, IBM Research, Zurich, Switzerland, 2016.
[19] S. De Angelis, A. Leonardo, B. Roberto, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain," in *Proceedings of the Italian Conference on Cyber Security*, p. 11, 2018, https://eprints.soton.ac.uk/415083/.
[20] A. Tenorio-Fornés, V. Jacynycz, D. Llop-Vila, A. Sánchez-Ruiz, and S. Hassan, "Towards a Decentralized Process for Scientific Publication and Peer Review Using Blockchain and IPFS," in *Proceedings of the Hawaii International Conference on System Sciences*, Maui, HI, USA, 2019.
[21] F. C. Coelho and A. Brandão, "Decentralising scientific publishing: can the blockchain improve science communication?" *Memórias do Instituto Oswaldo Cruz*, vol. 114, Article ID e190257, 2019.
[22] E. Stojmenova Duh, A. Duh, U. Droftina et al., "Publish-and-Flourish: using blockchain platform to enable cooperative scholarly communication," *Publications*, vol. 7, p. 33, 2019.
[23] A. C. Kade Morton, "Aletheia: blockchain for scientific knowledge with a community management framework," 2017, https://github.com/aletheia-foundation/aletheia-whitepaper/.
[24] A. Anita, A. Flora, C. Giovanni, G. Francesco, I. Nicla, and M. Antonino, "A study on textual features for medical records classification," *Studies in Health Technology and Informatics*, pp. 370–379, 2014.
[25] Ethereum Clients, http://www.ethdocs.org/en/latest/ethereum-clients/index.html.
[26] JSON RPC, https://github.com/ethereum/wiki/wiki/JSON-RPC#json-rpc-endpoint.
[27] N. Prusty, *Building Blockchain Projects: Building Decentralized Blockchain Applications with Ethereum and Solidity*, Packt Publishing Ltd., Birmingham, UK, 2017, ISBN 978-1-78712-214-7.
[28] L. Zhou, M. Marsh, F. Schneider, and A. Redz, *Distributed Blinding for Distributed ElGamal Re-Encryption*, IEEE, Columbus, OH, USA, 2005.