SciBlock: A Blockchain-Based Tamper-Proof Non-Repudiable Storage for Scientific Workflow Provenance

Dinuni Fernando [1], Siddharth Kulshrestha [1], J. Dinal Herath [1], Nitin Mahadik [1], Yanzhe Ma [1], Changxin Bai [2]
Ping Yang [1], Guanhua Yan [1], Shiyong Lu [2]

[1]*Department of Computer Science, State University of New York at Binghamton*
[2]*Department of Computer Science, Wayne State University*

***Abstract*—Modern scientific workflow systems lack strong support for protecting the scientific data and their provenance from being forged or altered. As a result, scientists may be misled into believing that they have found a specific result, but only to discover later that the data they used have been altered and should not be trusted. To address this limitation, we develop a new system called *SciBlock* that leverages recent advances in blockchain technology to provide a tamper-proof and non-repudiable storage for scientific workflow provenance. SciBlock provides primitives that allow users to query scientific workflow provenance data efficiently. Moreover, SciBlock offers the capability of invalidating wrong or outdated scientific workflow provenance data without removing them from the blockchain. We conducted extensive experiments to evaluate the performance and scalability of SciBlock. Our experimental results show that SciBlock offers a promising approach to enhancing scientific research integrity in a distributed collaborative environment.**

I. Introduction

Scientific workflows are developed for collaborative research projects that involve multiple geographically distributed organizations. Sharing of large datasets and computational methods across different administrative domains is critical for scientific collaborations in many areas such as high energy physics and bioinformatics.

The trustworthiness of scientific discoveries relies on the integrity of the data processed by scientific workflows and their underlying cyberinfrastructure. The lack of effective mechanisms for protecting the data integrity in modern scientific workflow systems may lead to undesirable scientific frauds or disputes: a scientist or a graduate student may forge/alter datasets, results, or computation to get papers published; a disgruntled member of a research group may forge data or modify the specification of the group's workflow to distort the output of the workflow; a malicious user may also forge a piece of data, post the data on a website, and claim that the data is produced by a scientific workflow. As a result, scientists may be misled into believing that they have found a specific result, but only to discover later that the data they used has been altered and should not be trusted.

In this work we develop a new system called *SciBlock* to provide a tamper-proof and non-repudiable storage for scientific workflow provenance data in a distributed collaborative environment. The provenance of a scientific workflow captures the derivation history of a data product and is hence essential to the reproducibility of scientific discovery [44]. To overcome the difficulty of establishing trust relationships in distributed environments, SciBlock leverages recent advances in blockchain technology to ensure the data processing integrity of scientific workflows. Built upon the distributed consensus, transaction verification, and record immutability features of blockchain, SciBlock offers the capabilities of verifying that a data product was generated by a specific workflow, querying how a data product is derived from a scientific workflow, and checking whether a specific scientific result is still valid.

The application of blockchain in SciBlock entails transformation from logs produced by scientific workflows that contain provenance information to transactions that are amenable to blockchain processing. We, however, cannot process scientific workflow provenance as blockchain transactions in a straightforward manner due to the following challenges. *Firstly*, once a provenance record is added to a blockchain, it is impossible to revoke the change, assuming that at least 51% of the participating peers would not collude to subvert its operation [10]. However, a scientific workflow may be modified frequently to fix bugs or add new features, and hence some of the data products generated previously may become invalid later. As a result, it is necessary to invalidate such data products in the blockchain. In this work, we propose a new mechanism to invalidate data products without deleting them from the blockchain. *Secondly*, blockchain allows users to verify the existence of a transaction efficiently, but offers no or limited capabilities for querying transactions by specific fields. Computing the derivation history of a data product in a large scientific workflow requires efficient queries of provenance records by their output fields. SciBlock addresses this issue by an off-chain approach that uses a combination of query on unencrypted database and blockchain verification to speed up the query process, while ensuring the trustworthiness of the query results. SciBlock's off-chain approach differs from the existing off-chain approaches in that SciBlock not only verifies the off-chain provenance records, but also the derivation history computed from the off-chain records. Because the database may be modified or deleted by unauthorized users and multiple derivation graphs may be computed for a specific scientific result (due to the execution of a scientific workflow multiple times), it is non-trivial to verify the correctness of the derivation history computed from the database.

In summary, our main contributions are as follows:

- We propose an off-chain approach that augments the blockchain-based transaction verification with queries of local auxiliary databases to expedite blockchain queries and data derivation history computation, and formally

978-1-7281-6739-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CIC48465.2019.00019

prove the correctness of this approach.

- We propose an efficient mechanism to invalidate wrong or outdated provenance records without removing them from the blockchain. Because our invalidation mechanism enables SciBlock to keep a complete history of workflow modifications, it can be used to prevent dishonest researchers' attempts of altering or forging scientific results.
- We have implemented SciBlock on top of the private Ethereum blockchain, a popular generic open-source blockchain platform, and conducted extensive experiments to evaluate the performance and scalability of SciBlock. Our experimental results show that our off-chain approach significantly reduces the query time and the time taken to compute the derivation history.

The rest of the paper is organized as follows. Section II presents related work. Section III provides an overview of scientific workflows, provenance, and blockchain. The design of SciBlock is given in Section IV. Sections V and VI present techniques for data derivation history computation and provenance invalidation, respectively. Section VII provides the implementation details of SciBlock. Our experimental results are given in Section VIII. Section IX concludes the paper.

II. Related work

Blockchain technology has been applied in a number of domains, including finance [47], banking [50], insurance [37], healthcare [9], etc.

A number of researchers have proposed techniques to secure provenance systems with centralized storage (e.g.[48], [41], [17], [34], [14], [51], [33]). In such systems, if the central server is compromised, then the entire data trail cannot be trusted anymore.

There is limited work in the investigation of the blockchain as a platform to secure scientific workflow provenance. Works in [13], [20], [27], [29] emphasize the importance of protecting provenance in scientific research and point out the potential applicability of blockchain as an enabler for creating such a platform. Ramachandran et al. [40] proposed a blockchain based scientific system called SmartProvenance, which automatically verifies the provenance records by utilizing a voting mechanism. DataProv [39] is a distributed system that securely captures the scientific data, which again uses a voting based mechanism to obtain the approval before storing data entries to the platform. ProvChain [31], [42] is a blockchain based data provenance system that provides assurance for cloud storage applications. DroneChain [32] is a public blockchain platform for securing data with the limited battery and process capability of drones. Tosh et al. [46] described design challenges and opportunities in developing proof-of-stake for data provenance in cloud platform. [38] uses a private blockchain based network to support data accountability and provenance tracking for European union residents' data. Brooks et al. [49] presented a blockchain system that can be used to secure data provenance outside users control. BlockFlow [18] is a workflow provenance system built on top of E-Science ECOsystem and ProvHL [19] is a provenance metadata storage system built on top of Hyper-ledger Fabric [8]. The above works neither use off-chain approaches to improve the efficiency of blockchain query nor consider the invalidation of provenance data.

Chen et al. [15] proposed CertChain, a public audit scheme for TLS connections based on the blockchain technology. The work in [16] uses blockchain to share scientific workflow provenance. The above works store the data products, but not the provenance, off-chain. MultiChain [24] uses an off-chain hashing solution to improve the scalability of the blockchain. [21] describes two ways to extend the smart contracts with off-chain logic and [22] discusses five off-chain patterns. In [26], the authors investigated the potential of increasing the scalability of the blockchain through off-chain storage and computation. Bitcoin lightning network [28] is an instant, high-volume micro-payment system operating outside the blockchain. SciBlock's off-chain approach differs from the above approaches in that SciBlock not only verifies the off-chain provenance records, but also the derivation history computed from the off-chain records.

Sigurjonsson et al. [43] propose to use blockchain and hash to protect the integrity of the workflow provenance and use InterPlanetary File System protocol for provenance version control. Although they also store the provenance both off-chain and on-chain, the execution time is stored only in the local database. Thus, if the local database is tampered, the database cannot be recovered from the blockchain. They also propose to use hash to protect the integrity of workflow provenance stored in the local database once the workflow completes the execution. This approach, however, does not prevent the provenance from being modified before a workflow finishes the execution. In addition, their version control prevents provenance records from being sent to other nodes in blockchain, but SciBlock's invalidation mechanism does not.

III. Background

This section provides an overview of scientific workflows and workflow provenance. As blockchain is the main methodology used in this work to ensure the integrity of workflow provenance, we also present a brief introduction to it.

A. Scientific Workflows and Provenance

Scientific workflow is a popular cyberinfrastructure paradigm to accelerate scientific discoveries and facilitate collaboration between geographically distributed organizations. Figure 1 shows a scientific workflow for performing intragenomic gene conversion analysis [7]. Each workflow task ($T_1 - T_7$) represents an individual computational step. The workflow takes as its input the protein sequences of a given genome and identifies all its multi-gene families (task T_1). A multi-gene family is then selected by the user and its associated DNA sequences are retrieved (task T_2). Next, a recombination analysis is performed on the retrieved sequences (task T_3) in two steps: a multiple DNA sequence alignment step (task T_4) and a gene conversion detection step (task T_5). The latter is implemented by GENECONV, an off-the-shelf program (task T_7), with an input data file preparation step (task T_6). A scientific workflow may be executed by multiple

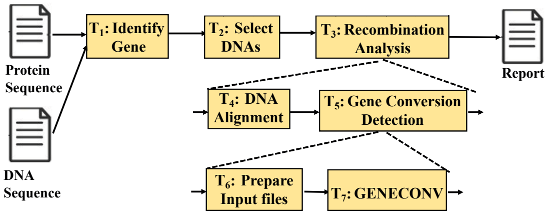


Fig. 1. A Gene Conversion Analysis Workflow.

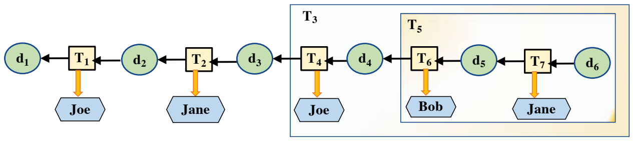


Fig. 2. A sample workflow provenance.

geographically distributed organizations. For example, tasks T_1 and T_2 may be executed by organization O_1 at location L_1 while task T_3 by organization O_2 at location L_2.

The *provenance* of a scientific workflow captures the derivation steps of a data product over a set of computational tasks. Figure 2 gives a sample provenance for the workflow in Figure 1, which is represented in a notation similar to the Open Provenance Model [4]. Circles represent data products, rectangles represent tasks, octagons represent users performing the tasks, and edges represent dependency relationships. Edge $d_i \leftarrow T_j$ specifies that data product d_i is the input to task T_j and edge $T_j \leftarrow d_i$ specifies that d_i is the output of task T_j. Edge $u \Leftarrow T_j$ specifies that task T_j was executed by user u.

B. Blockchain

Blockchain was first proposed in Bitcoin [36] to protect against double spending and modification of transactions. A blockchain consists of one or more blocks. Each block contains a block header and a number of Bitcoin transactions. To prevent unauthorized modification or forgery of blocks, each block in the blockchain is linked to its previous blocks by storing the hash of its parent block header. The Bitcoin blockchain uses Merkle trees [11] to efficiently verify whether a Bitcoin transaction exists in a block. The Merkle tree is created by repeatedly hashing pairs of transactions until it reaches the Merkle Root (i.e., the root hash). A Merkle tree generated from four transactions A, B, C, and D is given in Figure 3. When a Bitcoin transaction is added to the blockchain, the user receives a receipt from the blockchain which consists of the Merkle root and all hashes needed to verify the transaction. For example, to verify that transaction B exists in the Merkle tree in Figure 3, the receipt would contain the Merkle root (i.e. Hash ABCD), Hash A, and Hash CD. If Hash(Hash(Hash A, Hash(B)), Hash CD) is equal to the Merkle root, then the transaction exists in the blockchain.

The blockchain is a distributed ledger, in which the data is distributed across peer-to-peer networks to avoid a center point for attackers to corrupt the blockchain. Each peer in the peer-to-peer network has its own copy of a blockchain and all copies of the blockchain are synchonized across the network.

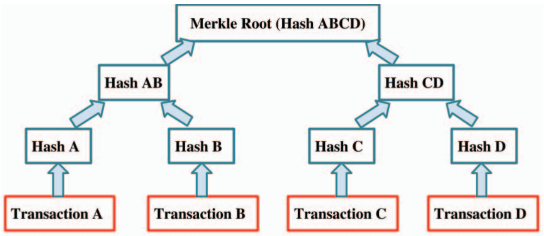


Fig. 3. An example Merkle tree.

IV. Design of SciBlock

This section presents the design of SciBlock. As mentioned in Section III.A, scientific workflow tasks may be executed by multiple organizations. Therefore, the integrity of a scientific workflow is ensured collectively by the integrity of all individual computational tasks, assuming that the communication channels among all the tasks are tamper resilient, e.g. protected by network security protocols such as SSL/TLS and IPSec.

SciBlock is designed as a cyberinfrastructure shared by geographically distributed organizations that collaboratively execute a scientific workflow, to provide integrity assurance of scientific workflow provenance. It is suitable for loosely connected collaborative research environments where there lacks a universally trusted authority to validate the integrity of scientific workflows (which is often the case in practice). Towards this end, SciBlock includes key functionalities to verify that a data product was generated from a specific workflow, examine the derivation history of scientific results, and verify that a scientific result is valid.

The architecture of SciBlock is illustrated in Figure 4. SciBlock is built upon a permissioned blockchain network with *proof of authority (POA) consensus* whose nodes are distributed across multiple sites. The pre-authenticated nodes in this permissioned blockchain network are contributed by various scientific organizations, such as universities and national laboratories, who have the incentive to ensure that their research is credible. Although pre-authentication introduces additional operational overhead, it circumvents the difficulty of designing both the costs for researchers to submit their provenance records and the incentives for miners who validate these records if we use a public permissionless blockchain to build SciBlock.

To prevent researchers from cheating, SciBlock stores scientific workflow provenance data represented as *provenance records* in blockchain. As shown in Table I, each provenance record contains a subset of provenance information related to a workflow task, including the task ID, the input to the task, the output generated by the task, the execution time, and the user who executed the task. We store the hash and the path of the input and output data in provenance records, instead of the real data, to reduce the size of provenance records. By computing the hash of a piece of data and comparing it against the hash stored in the corresponding provenance record, we can verify the integrity of the data.

The current prototype of SciBlock uses the DATAVIEW,

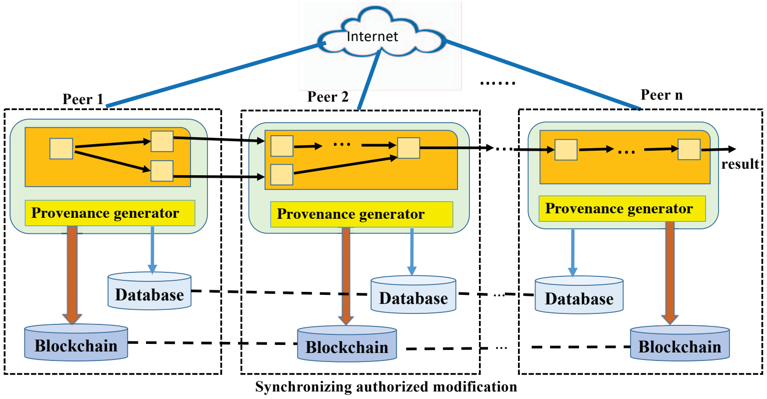


Fig. 4. The architecture of SciBlock.

Field	Description
task	The ID of the workflow task executed
input	Input to the workflow task (path and hash)
output	Output generated by the workflow task (path and hash)
time	Execution time
user	The person who executed the workflow task

TABLE I
THE PROVENANCE RECORD.

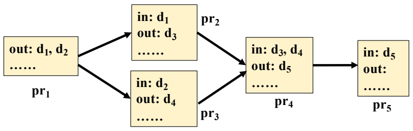


Fig. 5. Derivation graph.

a big data workflow management system [30], to execute scientific workflows. The provenance record generator generates provenance records during the execution of a workflow, and adds provenance records to the blockchain to prevent the records from being altered or deleted. The provenance records are also added to locally maintained databases, which are used for improving the efficiency of blockchain query (discussed in Section V). Although each peer in a private network has its own copy of the blockchain, any of its modification to the blockchain is synchronized automatically among all peers. When a data product generated by a workflow is not valid any more (e.g. the workflow task that generated the data has bugs), SciBlock adds an invalidation transaction to the blockchain and the invalidation module invalidates the corresponding provenance records. Users query the derivation history of scientific results through an HTML web interface. The derivation history generator computes the derivation history and returns a derivation graph to the user. The derivation history generator also informs the user if any data in the graph is invalid.

In a nutshell, the characteristics of SciBlock as described above can be summarized as follows:

- **Distributed consensus:** There is no centralized entity in SciBlock to synchronize the efforts of collecting provenance records submitted by different entities. Instead, the blockchain allows these geographically distributed entities to agree upon what provenance records have already taken place at any time point.
- **Tamper-proof:** Once a provenance record has been validated and added to the blockchain, it is impossible to revoke the change, assuming that at least 51% of the participating nodes in SciBlock would not collude to subvert its operation [10].
- **Non-repudiation:** SciBlock is built upon a permissioned Ethereum network with POA consensus, inside which each participating peer has a private-public key pair. Each individual provenance record is signed by the private key of the entity who has executed the corresponding workflow task. Hence, the execution of a workflow task cannot be repudiated by this entity later, assuming that its private key is secure.
- **Efficiency:** Due to the inefficiency of querying provenance records by their fields within a large blockchain, SciBlock augments the blockchain with locally maintained databases to expedite transaction queries while still ensuring the aforementioned properties.

Our threat model assumes that the provenance generator and data transmission channels, from the place where the workflow tasks are executed to the place where the provenance records are submitted, are protected using the existing security mechanisms (i.e., the provenance information is not tampered before it is submitted to SciBlock). Using the provenance information contained within provenance records, we can develop a variety of high-level functionalities that can help scientists to verify the integrity of scientific findings or revise existing ones. In this work we focus on two useful operations:

- **Computation of derivation history**: A scientist can use SciBlock to figure out how a data product – which can be an interesting scientific finding – is derived from the raw data available to the scientific community.
- **Invalidation of provenance records**: Modification of a workflow task would invalidate all the provenance records it has produced.

Sections V and VI elaborate on the above two functionalities, respectively.

V. Computing Derivation History

SciBlock offers scientists the capability of querying the derivation history of a scientific result, which enables scientists to verify the correctness of the result and to reproduce the result. The derivation history of a data product is represented as a *derivation graph*. Let pr be a provenance record. We use $pr.task$, $pr.input$, $pr.output$, $pr.user$, and $pr.time$ to represent the task, input, output, user, and time stamp fields in pr, respectively. Below, we define the derivation graph.

Definition 1 (Derivation/Derivation Graph): Let pr_1 and pr_2 be two provenance records. pr_2 is derived from pr_1, denoted as $pr_1 \rightarrow pr_2$, if there exists $in \in pr_2.input$ such that $in \in pr_1.output$. A derivation graph $G = \langle V, E \rangle$ is a directed acyclic graph (DAG), where each node $pr \in V$ is a provenance record and each edge $e \in E$ is a derivation of the form $pr_1 \rightarrow pr_2$. □

An example derivation graph is given in Figure 5. In this graph, pr_2 is derived from pr_1 because $d_1 \in pr_1.output$ and $d_1 \in pr_2.input$.

Computing the derivation history of a data product in a large scientific workflow requires efficient queries of transactions by their output fields. We implemented two primitives for querying provenance records by fields – one uses the brute force search and the other is based on the Bloom filter [1], a probabilistic data structure developed to improve the efficiency of search (details are given in Section VII). Our experimental results show that it takes $23 - 30$ms and $20 - 27$ms to query one transaction with the brute force search and the Bloom-filter based search, respectively.

In this section, we propose to use database query combined with blockchain verification to expedite the query process, while ensuring the trustworthiness of the query result. The provenance records are stored in both blockchain and locally maintained databases (called *provenance database*). Each provenance record stored in the database consists of all fields in Table I, as well as a "valid" field, which specifies whether the output data in the record is valid. The provenance database is protected by existing authentication and access control mechanisms, but is neither encrypted nor hashed in order to enable efficient query. Therefore, the provenance records stored in the database may be deleted or altered. As a result, for every provenance record returned from a database query, we verify that the record exists in the blockchain (using Ethereum API *getTransaction*). In the rest of the paper, we use the term *blockchain verification* to represent the process of checking whether a provenance record returned from a database query exists in the blockchain. Our current prototype of SciBlock uses SQLite and MySQL as provenance databases. Our experimental results show that querying SQLite (MySQL) combined with blockchain verification is about 7× (3.5×) faster than querying the Ethereum blockchain directly.

A derivation graph is either a *complete* or a *partial* graph, as defined below. We use the term *workflow input* to represent the data that is an input to a workflow task, but not an output of any workflow task (i.e. raw data or data generated from another workflow).

Definition 2 (Complete/Partial Node): A node v in a derivation graph $G = \langle V, E \rangle$ is a *complete node* if for every $in \in v.input$, in is either a workflow input or there exists $u \in V$ such that $in \in u.output$. A node v is a *partial node* if v is not a complete node. □

Definition 3 (Complete/Partial Derivation Graph): A derivation graph G is a *complete derivation graph* iff all nodes in G are complete nodes. A derivation graph G is a *partial derivation graph* iff G is not a complete derivation graph. □

A partial derivation node/graph is computed either because not all provenance records generated are added to the blockchain/database or because some provenance records in the database were deleted/altered by unauthorized users.

Algorithm 1 Computing the derivation graph

```
1:  procedure $derive\_history(pr)$
2:  if $pr$ does not exist in the blockchain then
3:      print "error: $pr$ does not exist in the blockchain"; return;
4:  if all inputs in $pr$ are workflow inputs then
5:      return $\langle \{pr\}, \emptyset \rangle$;
6:  $V = workset = \{pr\}$; $E = list = \emptyset$
7:  while $workset \neq \emptyset$ do
8:      remove $w$ from $workset$;
9:      for every $in \in w.input$ do
10:         if there exists $pr_1$ such that $in \in pr_1.output$ then
11:             if $pr_1$ exists in the blockchain then
12:                 $E = E \cup \{pr_1 \rightarrow w\}$;
13:                 if $pr_1 \notin V$ then
14:                     $workset = workset \cup \{pr_1\}$;
15:                     $V = V \cup \{pr_1\}$;
16:             else
17:                 Print "error: database and blockchain are inconsistent"; return
18:         else
19:             if $in$ is not a workflow input then
20:                 add $in$ to $list$;                ▷ partial graph
21: if $list == \emptyset$ then return $\langle V, E \rangle$;
22: for every provenance record $pr$ in blockchain do
23:     if there exists $out \in pr.output$ such that $out \in list$ then
24:         Print "error: database and blockchain are inconsistent"; return;
25: return $\langle V, E \rangle$;
```

Algorithm 1 gives the pseudocode for computing the derivation history of a provenance record pr. The algorithm first checks whether pr is present in the blockchain, and if not, the algorithm reports error (lines 2– 3). If pr is present in the blockchain and all inputs in pr are workflow inputs (which means that tr is not derived from any other provenance records), then the algorithm returns a graph containing one node pr (lines 4–5). Otherwise, the algorithm computes the derivation graph for pr from the provenance database (lines 6 - 15). If a complete derivation graph is computed and all nodes in the graph are present in the blockchain, then the algorithm returns the graph (line 21). Otherwise, for each input of partial nodes that has no derivation (stored in $list$), the algorithm checks whether a derivation can be computed for the input in the blockchain (line 23). This is used to ensure that the database is not modified. If such a derivation exists, then the algorithm reports inconsistency between the database and the blockchain (lines 24); otherwise, the algorithm returns the graph computed (line 25).

We use $G(pr)$ to represent the derivation graph computed for provenance record pr. Theorems 1 and 2 prove the soundness and the completeness of Algorithm 1, respectively.

Theorem 1 (Soundness): Let pr be a provenance record. If Algorithm 1 returns a derivation graph $G(pr)$, then $G(pr)$ can be computed from the blockchain.

Proof: If Algorithm 1 returns a derivation graph $G(pr)$, then for every edge $pr_1 \rightarrow pr_2$ in $G(pr)$, there exists $in \in pr_2.input$ such that $in \in pr_1.output$. Because the algorithm returns $G(pr)$, according to line 11 of the algorithm, pr_1 and pr_2 are present in the blockchain. As there exists $in \in pr_2.input$ such that $in \in pr_1.output$, $pr_1 \rightarrow pr_2$ can be computed from the blockchain as well. The theorem holds. □

Theorem 2 proves the completeness of Algorithm 1, which states that for every derivation graph $G(pr)$ computed from the blockchain, Algorithm 1 either returns an equivalent derivation graph (defined below) or reports inconsistency between the database and the blockchain. Note that Theorem 2 does not guarantee that for every derivation graph computed from the blockchain, the same graph can be computed from the provenance database. As a workflow task may be executed multiple times, multiple provenance records may have the same input, task, and output fields but different execution times. Therefore, multiple derivation graphs may be computed for a provenance record. Algorithm 1 returns only *one* of the derivation graphs. It is possible that some provenance records have been deleted from the database, but the deleted records do not affect the derivation graph returned from the database. Below, we define equivalent derivation graphs and proves Theorem 2.

Definition 4 (Equivalent Provenance Records/Derivation Graphs): Two provenance records pr_1 and pr_2 are equivalent, denoted as $pr_1 \equiv pr_2$, iff $pr_1.input = pr_2.input$, $pr_1.output = pr_2.output$, and $pr_1.task = pr_2.task$. Two derivation graphs $G(pr)$ and $G'(pr)$ are equivalent, denoted as $G(pr) \equiv G'(pr)$, iff for every edge $pr_1 \rightarrow pr_2$ in $G(pr)$, there exists an edge $pr'_1 \rightarrow pr'_2$ in $G'(pr)$ such that $pr_1 \equiv pr'_1$ and $pr_2 \equiv pr'_2$, and vice versa. □

As each workflow task is deterministic, two provenance records have the same output only if they were generated from the same workflow task with the same input. Therefore, all derivation graphs computed for a specific provenance record are equivalent.

Theorem 2 (Completeness): Let pr be a provenance record. If a derivation graph $G(pr)$ can be computed from the blockchain, then Algorithm 1 either returns a graph $G'(pr)$ such that $G'(pr) \equiv G(pr)$ or reports error.

Proof: The theorem is proved by induction on the number of nodes in the derivation graph.

Base case: Assume that $G(pr)$ consists of one node pr. Then either (1) all inputs in pr are workflow inputs, or (2) for all $in \in pr.input$ that are not workflow inputs, there does not exist pr' in the blockchain such that $in \in pr'.output$. In case (1), Algorithm 1 returns $G(pr)$. In Case (2), if the database is consistent with the blockchain, then Algorithm 1 returns $G(pr)$. Otherwise, if there exists $in \in pr.input$ and pr' in the database such that $in \in pr'.output$, then because pr' does not exist in the blockchain, Algorithm 1 returns an error (lines 16–17). The theorem holds.

Induction: Assume that the theorem holds for all derivation graphs that have $\leq k$ nodes. We now prove that the theorem holds for derivation graphs that have $k+1$ nodes. Let $G(pr)$ be a derivation graph computed from the blockchain that contains $k+1$ nodes and pr_2 be the last node computed in this graph. We now prove that if $pr_2 \rightarrow pr_1$ can be computed from the blockchain, then there exist pr'_2 and pr'_1 such that $pr'_2 \rightarrow pr'_1$ can be computed from the database and $pr_1 \equiv pr'_1$ and $pr_2 \equiv pr'_2$. We prove the theorem by contradiction. Assume that such an edge cannot be computed from the database. By induction hypothesis, there exists pr'_1 in the database such that $pr'_1 \equiv pr_1$. Because the edge cannot be computed from the database, there does not exist a provenance record pr'_2 in the database such that $pr'_2 \equiv pr_2$. Therefore, pr'_1 is a partial node in the derivation graph computed from the database. Let $d \in (pr_2.out \cap pr_1.in)$. Line 23 of Algorithm 1 would check whether there exists a provenance record pr in the blockchain such that $pr.out = d$. Because $pr_2.out = d$ and pr_2 is in the blockchain, the algorithm reports an error (line 24) instead of returning a derivation graph. This contradicts to the assumption. Therefore, the theorem holds. □

VI. Provenance Invalidation

Once a provenance record is added to the blockchain, it is impossible to revoke the change, assuming that the honest nodes together possess more than half of the mining power. A workflow task may be modified to fix bugs or add new functionalities. When a workflow task is modified, the data previously generated from this task and all data derived from it are not valid any more and hence need to be invalidated. To address this issue, this section presents a novel technique to enable users to invalidate wrong or outdated provenance records without removing them from the blockchain. SciBlock currently supports the invalidation of provenance records generated prior to specific time. For example, if a workflow task is modified, then the scientists can re-run the workflow and invalidate all previous provenance records.

We use invalidation transactions to invalidate provenance records. The invalidation transaction contains one field $time$; provenance records with an execution time prior to $time$ will be invalidated. When an invalidation transaction is added to a blockchain, the invalidation module adds the hash of all provenance records invalidated by this transaction to the blockchain. As an invalidation transaction is often submitted after a task is modified to generate new data, to prevent users from mistakenly adding an invalidation transaction, we provide an option for the user to check whether a corresponding new data product has been generated for each invalidated data product. If the invalidation condition does not hold, then the user decides whether he/she wants to proceed.

Algorithm 2 gives the pseudocode for invalidating provenance records. Procedure $check\_invalidate$ returns the task field ($tasklist$) of all provenance records that have an execution time later than $time$. Procedure $invalidation$ invalidates transactions. If the invalidation condition is turned off, then the algorithm invalidates the corresponding provenance records (lines 2–5). Otherwise, for every provenance record whose execution time is earlier than $time$, the algorithm checks whether its task field is in $tasklist$ and if so (i.e. the invalidation condition holds), invalidates the record (lines 8 – 11); otherwise, the algorithm asks the user whether she wants to invalidate the record. To improve the efficiency, the algorithm first checks whether the invalidation condition holds in the provenance database and then verifies the result in the blockchain.

Algorithm 2 Transaction Invalidation

```
procedure $invalidation(time, opt)$
if $opt == 0$ then
    for every $pr$ in blockchain where $pr.time < time$ do
        invalidate $pr$ in blockchain;
        set $pr.valid$ to be 0 in DB;
else
    $tasklist = check\_invalidate(time)$;
    for every valid record $pr \in$ blockchain do
        if $(pr.time < time$ and $pr.task \in tasklist)$ then
            invalidate $pr$ in blockchain;
            set $pr.valid$ to be 0 in DB;
        else
            if $pr.time < time$ and there exists $pr_1 \in blockchain$ such that $pr_1.task = pr.task$ and $pr_1.time > time$ then
                print "Inconsistency between database and blockchain"; return error;

procedure $check\_invalidate(time)$
$tasklist = \emptyset$;
for every $pr$ in DB such that $pr.time > time$ and $pr \in blockchain$ do
    if $pr.task \notin tasklist$ then $tasklist = tasklist \cup \{pr.task\}$;
return $tasklist$;
```

Restoring the provenance database: If any of the algorithms reports an error, or if there are signs that the provenance database has been compromised, then the provenance database needs to be restored from the blockchain. The provenance database is restored as follows. For every provenance record in the blockchain, we check whether the record is valid. We then add the record and the validation status to the provenance database. The above approach requires to traverse the blockchain only once.

VII. Implementation of SciBlock

A number of blockchain platforms were developed including Ethereum [23], Tierion [45], Hyperledger [25], Bigchain [12], and MultiChain [35], etc. We chose to implement SciBlock on top of the Ethereum Parity [2] blockchain, because Parity is a permissioned blockchain with POA consensus that provides both authentication and tamper proof, and Parity has better performance than Ethereum Geth.

Adding provenance records/invalidation transactions to blockchain: The format of the provenance records is specified using smart contracts in Ethereum and the provenance record is created using the *buildTransaction* API. The provenance records are submitted to private blockchains via a python web RPC interface called *web3.py*. When a user submits a provenance record, *web3.py* invokes the *sendTransaction()* RPC call to add the record to the blockchain. The information related to the provenance record is then anchored to the Merkle tree and a transaction receipt is returned to the user. Invalidation transactions are authenticated and added to the blockchain similarly.

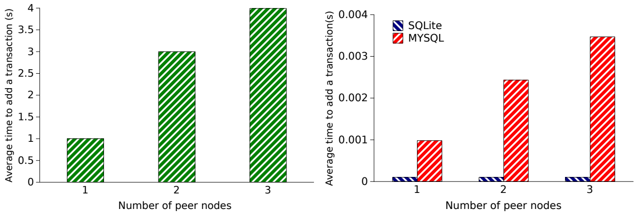
Querying provenance records by fields: We implemented two primitives for querying the provenance records by fields. (1) a naive primitive that uses brute-force search to locate a specific provenance record and (2) a primitive based on the Bloom filter [1], which is a probabilistic data structure developed to improve the efficiency of search. In the Bloom filter-based implementation, when a user submits a provenance record, the event/log mechanism in Ethereum captures and anchors the data as logs in the blockchain. The logs are stored in the *logsBloom* data structure in the block header of the Bloom filter, which consists of indexable information that utilizes storage efficiently. Ethereum Bloom filter has the following limitations. First, the index field can contain maximum of 32 bytes. However, each field of a provenance record is a string that can have arbitrary length. To counter this limitation, we converted the field that needs to be indexed into type *Bytes32* that represents 32-byte strings. Secondly, up to three fields can be indexed in the Bloom filter. Thirdly, the Bloom filter produces false positives and hence the result returned from the Bloom filter needs to be verified.

Presence of a provenance record in the blockchain: Authentication is not required to verify the existence of a provenance record in the blockchain. External users can submit query requests through web interface. SciBlock uses the *getTransactionReceipt()* RPC call to check whether a provenance record exists in the blockchain, which takes as input the hash of a provenance record and returns the corresponding receipt if the record exists in the blockchain.

Provenance invalidation: When an invalidation transaction is added to a blockchain, Sciblock computes all provenance records that are not previously invalidated and are invalidated by this transaction. Sciblock then stores the hash of each invalidated provenance record as a new transaction in the blockchain and indexes the hash field. This enables users to check if a provenance record is valid efficiently using the Bloom filter. Alternatively, instead of storing the hash of each invalidated provenance record, we can just store the invalidation transaction in the blockchain. When a user queries whether a provenance record is valid, we check whether the record is invalidated by any of the invalidation transactions in the blockchain. Compared to the approach used in our implementation, this approach has less invalidation time, but imposes higher performance overhead on checking whether a provenance record is valid.

Provenance database: SciBlock uses SQLite [5] and MySQL [3], two widely used database management systems, as our provenance databases. SQLite is an embedded database management system that has superior performance for single machine access, but is not intended to be used as client-server applications. As a result, SQLite is installed on each local machine and all copies of a SQLite database on different machines are synchronized through the network. MySQL is much slower than SQLite for single machine access, but is good for concurrent access by a large number of users/machines through the network.

To synchronize the SQLite databases on different ma-



(a) (b)

Fig. 6. Average time for adding one transaction from 1-3 peers on a single machine: (a)blockchain (b)databases.

chines, a separate client program executes on each node connects and synchronizes with its locally cached blockchain using the *web3* interface. When a new provenance record is added to the blockchain, the client obtains the corresponding block number using *web3.eth.blockNumber()* and compares it against the last synchronized block number. The client then uses *web3.eth.getTransactionFromBlock()* and *web3.eth.getTransactionReceipt()* to get the corresponding transaction receipt. The provenance logs are stored in the transaction receipt in an hex encoded format. The client program then decodes the logs to obtain the provenance record and adds the record to the provenance database. The above process synchronizes SQLite databases and the blockchain within a time frame of approximately 70ms for each provenance record.

VIII. Evaluation

We evaluate the performance of SciBlock using a real workflow provenance generated by the DATAVIEW workflow management system [30] and synthetic benchmarks generated by a workflow provenance generator written by us. All experimental results were obtained on dual two-core 3.30 GHz Intel Xeon machines with $8GB$ memory connected through a Gigabit Ethernet switch with 1 Gbps full-duplex ports.

A. Experimental Results: Single Machine

In Ethereum, users can create multiple peers on a single machine. Each peer takes users' input and performs the functionality defined in the smart contracts. This section presents the performance results of SciBlock on a single machine with 1-3 peers. We installed Ethereum parity, SQLite, and MySQL on each of the machines used in our experiments. The blockchain is not synchronized among different machines in our single-machine experiments. We have also installed MySQL on an external machine in the same local-area network to enable remote access to MySQL through the network.

Average time taken to add one transaction: Figure 6(a) gives the average time taken for adding one provenance record to the blockchain and databases from 1-3 peers on a single machine. The number of provenance records added varies between 2000 and 10000. Each node added $total\_record/x$ provenance records to the blockchain/databases simultaneously, where $total\_record$ is the total number of records added $(2000-10000)$ and x is the number of peers $(1-3)$. The figure shows that the time taken for adding one provenance record to the blockchain increases when the number of peers increases. In addition, with the same number of peers, the time taken for adding one provenance record is constant, irrespective to the size of the blockchain. Adding one provenance record to databases is $1000-10000$ times faster than adding one record to the blockchain (as shown in Figure 6(b)).

Average query time: Figure 7 gives the average time taken for querying one provenance record by the output field from a single peer on a single machine. The x-axis in the figure represents the size of the blockchain/databases (2000 – 10000 provenance records) and the y-axis represents the average query time, which was calculated as an average over ten randomly generated outputs. MySQL(L) and MySQL(G) in the figure represent querying MySQL on the local and remote machines, respectively. blockchain-Naive and blockchain-Bloom represent our brute-force query primitive and Bloom-filter based query primitive, respectively. Figure 7 shows that blockchain-Bloom is 10.7% faster than blockchain-Naive, and querying SQLite and MySQL databases is significantly faster than querying the blockchain in all experiments.

We have also measured the average time taken to query one provenance record from 2-3 peers on a single machine simultaneously. Our experimental results show that the average query time with 2 and 3 peers is $10-14\%$ higher than that with a single peer for both blockchain and databases.

Average blockchain verification time: The average time taken to check whether a provenance record is present in a blockchain is constant ($3ms$) for blockchains with 2000-10000 provenance records. This result and Figure 7 together show that, querying SQLite (MySQL) combined with blockchain verification is about 7 times and $3-3.5$ times faster than querying the blockchain directly.

Computing Derivation History: Figure 8 gives the time taken to compute the derivation graphs that contain a sequence of 2000-10000 nodes, from a **single** peer on a single machine. SQLite/MySQL+verify represents our implementation of Algorithm 1, which uses database query combined with blockchain verification to compute derivation graphs. The figure shows that the time taken to compute a derivation graphs increases when the graph size increases for both blockchain and databases. The figure also shows that SQLite performs best, followed by SQLite+verify. Blockchain-Bloom has the worst performance, which is 5.3 times worse than SQLite+verify and 1.9 times worse than MySQL+verify.

Figure 9 gives the time taken to compute the derivation graph for provenance records containing two inputs. The inputs to each record are randomly chosen from the outputs that have already been generated. Figure 10 gives the number of nodes and edges in the graphs computed. Similar to Figure 8, SQLite performs best and Blockchain-Bloom performs worst.

We have also collected provenance records produced from a diagnosis recommendation workflow [6] in DATAVIEW and computed the derivation graph for the output of the workflow. The derivation graph consists of 8 nodes and 9 transitions. The time spent in computing derivation graph is $1ms$ for SQLite+verification, 0.39 seconds for MySQL+verification, and 0.51 seconds for Blockchain-Bloom.

In addition, we have measured the time taken to compute

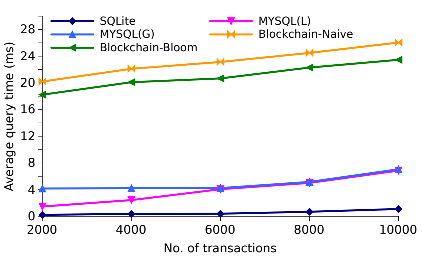


Fig. 7. Average time for querying one provenance record.

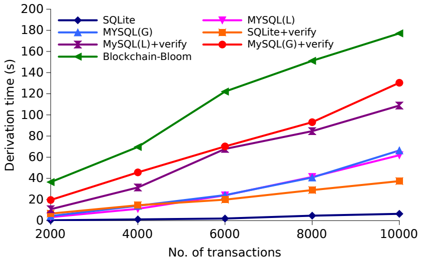


Fig. 8. Time for computing derivation graphs that contain 2000-10000 nodes.

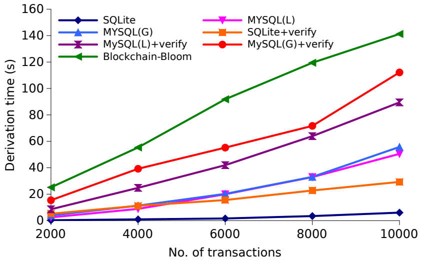


Fig. 9. Time for computing derivation graphs from provenance records with 2 inputs.

the derivation graph simultaneously from 2 and 3 peers. The experiments conducted are the same as those conducted for a single peer. Our experimental results show that the average time taken to compute the derivation graph is about the same for different number of peers.

Invalidation time: We have measured the average time taken to invalidate 2000-10000 provenance records in the blockchain. The time is constant (1 second), irrespective to the blockchain size. This is because every time a provenance record is invalidated, the hash of the record is added to the blockchain, which takes about 1 second. Adding hashes of invalidated records enables users to query whether a record is valid efficiently. Our experimental results show that, it takes about $16ms$ ($18ms$) to check whether a record is valid, when all records in the blockchain are valid (invalid). The average time taken to invalidate one provenance record from multiple peers is the same as that taken to add one record to the blockchain from multiple peers (shown in Figure 6).

Memory overhead: Figure 11 gives the memory usage for computing derivation graphs whose sizes vary between 2000 to 10000. The memory usage was captured using the *TOP* Linux utility. Compared to blockchain, SQLite with verification and MySQL with verification imposes $0.9\% - 8\%$ and $7\% - 18\%$ overhead on memory usage, respectively.

B. Experimental Results: Multiple Machines

In Ethereum, blockchains are distributed across peer-to-peer networks. In order to add transactions to the same blockchain from different machines, we need to synchronize peers on different machines. Each peer in Ethereum network is uniquely identified with a URL scheme called an "enode". Each enode consists of a hexadecimal node ID encoded with a username, an IP address, and a TCP listening port number. We maintain a list of peers participating in the network and the peer discovery protocol is used to connect peers in the network based on the list. Once the peers on different machines get connected, the peers start to synchronize with each other so that each machine has its own copy of the same blockchain. When a peer adds a provenance record to a blockchain, the record is also added to all other synchronized copies. Therefore, when a peer issues a query to the blockchain, the query is performed on its local copy of the blockchain and hence the query time is the same as that of a single machine. Similarly, the time spent in computing the derivation history and checking whether a provenance record is valid is the same as that of a single machine. As a result, this section reports only the time taken to add transactions from multiple machines.

Figure 12 gives the average time taken to add one provenance record to the same blockchain simultaneously from 1-3 machines (one peer runs on each machine). The figure shows that the average time increases from $1ms$ to $9ms$, when the number of machines increases from 1 to 3. The average time for adding one record from two (three) machines is $1.67x$ ($2.25x$) slower than adding one record from two (three) peers on a single machine, due to the synchronization between blockchain copies stored on multiple machines.

IX. Conclusion

This paper presents SciBlock, a system that leverages recent advances in the blockchain technology to provide a tamper-proof and non-repudiable storage for scientific workflow provenance data in a distributed collaborative environment. SciBlock enables scientists to verify the trustworthiness of scientific data and reproduce scientific results. SciBlock also offers the capability of invalidating wrong or outdated scientific workflow provenance data without removing them from the blockchain. We have conducted extensive experiments to evaluate the performance and the scalability of SciBlock. Our experimental results show that SciBlock offers a promising approach to enhancing scientific research integrity in a distributed collaborative environment.

Acknowledgement: This work is supported in part by the National Science Foundation under grant OAC-1738929. We thank Nikhil Raverkar and Wen Yang for their help in SciBlock implementation.

References

[1] Ethereum bloom filter. https://github.com/ethereum/eth-bloom.
[2] Ethereum parity. https://www.parity.io/.
[3] Mysql. https://www.mysql.com/.
[4] *Open Provenance Model*. http://twiki.ipaw.info/bin/view/Challenge/OPM.
[5] Sqlite. https://www.sqlite.org/index.html.
[6] I. Ahmed, S. Lu, C. Bai, and F. A. Bhuyan. Diagnosis recommendation using machine learning scientific workflows. In *IEEE International Congress on Big Data*, pages 82–90, 2018.
[7] J. Alhiyafi, C. Sabesan, S. Lu, and J. L. Ram. RECOMBFLOW: A scientific workflow environment for intragenomic gene conversion analysis in bacterial genomes. *International Journal of Bioinformatics Research and Applications*, 5(1):1–19, 2009.

No. of transactions	No. of nodes	No. of edges
2000	1531	1533
4000	3175	3177
6000	4794	4796
8000	6303	6305
10000	8093	8095

Fig. 10. The size of derivation graphs for provenance records with two inputs

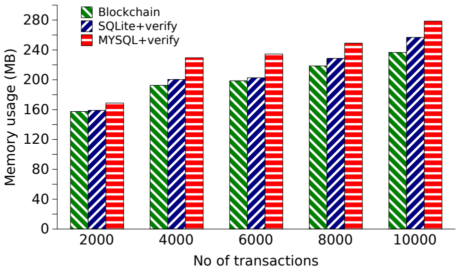


Fig. 11. Memory usage for computing derivation graphs from transactions with single inputs.

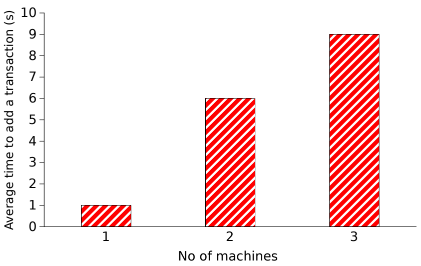


Fig. 12. Average time taken to add one transaction to the same blockchain from multiple machines.

[8] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *EuroSys*, 2018.

[9] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. Medrec: Using blockchain for medical data access and permission management. In *International Conference on Open and Big Data*, pages 25–30, 2016.

[10] A. Baliga. Understanding blockchain consensus models. *Persistent*, 2017.

[11] G. Becker and R. universitt Bochum. Merkle signature schemes, merkle trees and their cryptanalysis, 2008.

[12] BigchainDB : The Blockchain Database. https://www.bigchaindb.com /whitepaper/bigchaindb-whitepaper.pdf.

[13] J. J. Billings. Applying distributed ledgers to manage workflow provenance. *arXiv preprint arXiv:1804.05395*, 2018.

[14] A. Chebotko, S. Lu, S. Chang, F. Fotouhi, and P. Yang. Secure scientific workflow provenance querying with security views. *IEEE Transactions on Services Computing*, 3(4):322–337, 2010.

[15] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du. Certchain: Public and efficient certificate audit based on blockchain for tls connections. In *IEEE INFOCOM*, pages 2060–2068, 2018.

[16] W. Chen, X. Liang, J. Li, H. Qin, Y. Mu, and J. Wang. Blockchain based provenance sharing of scientific workflows. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3814–3820. IEEE, 2018.

[17] J. Cheney. A formal framework for provenance security. In *24th Computer Security Foundations Symposium*, pages 281–293, 2011.

[18] R. Coelho, R. Braga, J. David, F. Campos, and V. Stroele. Blockflow: Trust in scientific provenance data. In *Anais do XIII Brazilian e-Science Workshop*, pages 56–63, Porto Alegre, RS, Brasil, 2019. SBC.

[19] A. Demichev, J. Dubenskaya, E. Fedotova, A. Kryukov, S. Polyakov, and N. Prikhodko. Provenance metadata management in distributed storages using the hyperledger blockchain platform, 2019.

[20] A. Demichev, A. Kryukov, and N. Prikhodko. The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform. *arXiv:1811.12706*, 2018.

[21] L. Desrosiers and R. Olivieri. Extend your blockchain smart contracts with off-chain logic, 2018.

[22] J. Eberhardt and S. Tai. On or off the blockchain? insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*, pages 3–15. Springer, 2017.

[23] Ethereum blockchain platform. https://www.ethereum.org/.

[24] G. Greenspan. Scaling blockchains with off-chain data. https://www.multichain.com/blog/2018/06/scaling-blockchains-off-chain-data/.

[25] Hyperledger. https://www.hyperledger.org.

[26] IBM. Why new off-chain storage is required for blockchains document version 1.0, 2018.

[27] K. Janowicz, B. Regalia, P. Hitzler, G. Mai, S. Delbecque, M. Fröhlich, P. Martinent, and T. Lazarus. On the prospects of blockchain and distributed ledger technologies for open science and academic publishing. *Semantic Web*, pages 1–11, 2018.

[28] T. D. Joseph Poon. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, 2016.

[29] D. Karastoyanova and L. Stage. Towards collaborative and reproducible scientific experiments on blockchain. In *International Conference on Advanced Information Systems Engineering*, pages 144–149, 2018.

[30] A. Kashlev and S. Lu. A system architecture for running big data workflows in the cloud. In *Proc. Of the IEEE International Conference on Services Computing (SCC)*, pages 51–58, 2014.

[31] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *CCGRID*, pages 468–477, May 2017.

[32] X. Liang, J. Zhao, S. Shetty, and D. Li. Towards data assurance and resilience in iot using blockchain. In *IEEE Military Communications Conference*, pages 261–266, 2017.

[33] R. Luo, P. Yang, S. Lu, and M. I. Gofman. Analysis of scientific workflow provenance access control policies. In *The 9th IEEE International Conference on Services Computing (SCC)*, pages 266–273, 2012.

[34] P. McDaniel. Data provenance and security. *IEEE Security & Privacy*, 9:83–85, 2011.

[35] MultiChain Private Blockchain. https://www.multichain.com/download /multichain-white-paper.pdf.

[36] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[37] I. Nath. Data exchange platform to fight insurance fraud on blockchain. In *ICDM Workshop*, pages 821–825, 2016.

[38] R. Neisse, G. Steri, and I. N. Fovino. A blockchain-based approach for data accountability and provenance tracking. In *International Conference on Availability, Reliability and Security*, 2017.

[39] A. Ramachandran and M. Kantarcioglu. Using blockchain and smart contracts for secure data provenance management. *CoRR*, abs/1709.10000, 2017.

[40] A. Ramachandran and M. Kantarcioglu. Smartprovenance: A distributed, blockchain based dataprovenance system. In *ACM Conference on Data and Application Security and Privacy*, pages 35–42, 2018.

[41] A. Rosenthal, L. Seligman, A. Chapman, and B. Blaustein. Scalable access controls for lineage. In *workshop on Theory and practice of provenance*, page 110, 2009.

[42] S. Shetty, V. Red, C. Kamhoua, K. Kwiat, and L. Njilla. Data provenance assurance in the cloud using blockchain. volume 10206, pages 10206 – 10206 – 11, 2017.

[43] S. M. K. Sigurjonsson. Blockchain use for data provenance in scientific workflow. http://www.diva-portal.org/smash/get/diva2:1235451/ FULLTEXT02, 2018.

[44] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-Science. *SIGMOD Record*, 34(3):31–36, Sept. 2005.

[45] Tierion Platform. https://tierion.com/.

[46] D. Tosh, S. Shetty, X. Liang, and C. A. K. L. Njilla. Consensus protocols for blockchain-based data provenance: Challenges and opportunities. In *Ubiquitous Computing, Electronics and Mobile Communication*, 2017.

[47] P. Treleaven, R. G. Brown, and D. Yang. Blockchain technology in finance. *Computer*, 50(9):14–17, 2017.

[48] W. Tsai, X. Wei, Y. Chen, R. Paul, J. Chung, and D. Zhang. Data provenance in soa: security, reliability, and integrity. 1(4):223247, 2007.

[49] Using The Blockchain To Secure Provenance MetaData. https://archive.org/details/usingtheblockchaintosecureprovenance metadatatechnicalreport.

[50] T. Wu and X. Liang. Exploration and practice of inter-bank application based on blockchain. In *2017 12th International Conference on Computer Science and Education (ICCSE)*, pages 219–224, Aug 2017.

[51] P. Yang, X. Xie, I. Ray, and S. Lu. Satisfiability analysis of workflows with control-flow patterns and authorization constraints. *IEEE Transactions on Services Computing*, pages 237–251, 2014.