


## Article

# BSTProv: Blockchain-Based Secure and Trustworthy Data Provenance Sharing

Lian-Shan Sun <sup>1,2,\*</sup> , Xue Bai <sup>1,2</sup>, Chao Zhang <sup>1,2</sup>, Yang Li <sup>1,2</sup>, Yong-Bin Zhang <sup>1,2</sup> and Wen-Qiang Guo <sup>1,2</sup> 

- <sup>1</sup> School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China; safiyahcc@outlook.com (X.B.); sdtzcc@outlook.com (C.Z.); muziyangde@outlook.com (Y.L.); zhangyongbin@sust.edu.cn (Y.-B.Z.); guowenqiang@sust.edu.cn (W.-Q.G.)
- <sup>2</sup> Shaanxi Joint Laboratory of Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China
- \* Correspondence: sunlianshan@sust.edu.cn

**Abstract:** In the Big Data era, data provenance has become an important concern for enhancing the trustworthiness of key data that are rapidly generated and shared across organizations. Prevailing solutions employ authoritative centers to efficiently manage and share massive data. They are not suitable for secure and trustworthy decentralized data provenance sharing due to the inevitable dishonesty or failure of trusted centers. With the advent of the blockchain technology, embedding data provenance in immutable blocks is believed to be a promising solution. However, a provenance file, usually a directed acyclic graph, cannot be embedded in blocks as a whole because its size may exceed the limit of a block, and may include various sensitive information that can be legally accessed by different users. To this end, this paper proposed the BSTProv, a blockchain-based system for secure and trustworthy decentralized data provenance sharing. It enables secure and trustworthy provenance sharing by partitioning a large provenance graph into multiple small subgraphs and embedding the encrypted subgraphs instead of raw subgraphs or their hash values into immutable blocks of a consortium blockchain; it enables decentralized and flexible authorization by allowing each peer to define appropriate permissions for selectively sharing some sets of subgraphs to specific requesters; and it enables efficient cross-domain provenance composition and tracing by maintaining a high-level dependency structure among provenance graphs from different domains in smart contracts, and by locally storing, decrypting, and composing subgraphs obtained from the blockchain. Finally, a prototype is implemented on top of an Ethereum-based consortium blockchain and experiment results show the advantages of our approach.



**Citation:** Sun, L.-S.; Bai, X.; Zhang, C.; Li, Y.; Zhang, Y.-B.; Guo, W.-Q. BSTProv: Blockchain-Based Secure and Trustworthy Data Provenance Sharing. *Electronics* **2022**, *11*, 1489. <https://doi.org/10.3390/electronics11091489>

Academic Editor: George A. Tsihrintzis

Received: 11 April 2022

Accepted: 4 May 2022

Published: 6 May 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** blockchain; data provenance; secure and trustworthy data sharing; smart contract

## 1. Introduction

In the Big Data era, massive data are incessantly generated, shared, and used across organizations, and have shown their huge potential in changing the way of industrial production and social life [1]. The high trustworthiness of data is a precondition required for making dependable decisions. Erroneous, fabricated, or falsified data could lead to various undesired and unexpected aftermaths [2,3]. For example, in collaborative scientific research project, multiple distributed organizations independently conduct research activities and then share some datasets, computational methods, and research results with each other. If some of these datasets or results which are accidentally or intentionally fabricated or tampered, other organizations can be misled to make wrong or even harmful decisions that may in turn jeopardize economy or public security. How can we improve the trustworthiness of data shared across organizations? Data provenance is believed to be a promising solution [4].

According to W3C, data provenance usually records the entities, processes, or agents involved in producing a data object, and is usually structured as a directed acyclic graph,

called a provenance graph [5]. By analyzing the provenance graph of a data object, users can trace historical causes that make the data object become what it is; thus, the trustworthiness of the data object can be enhanced [4]. A provenance graph on the data object that has evolved for a long time may be very large and may contain various sensitive pieces of information [6]. It is neither efficient nor secure to share a provenance graph as a whole. A provenance management system should be able to flexibly provide users with secure and trustworthy provenance views, which are useful subsets of a provenance graph.

Nowadays, massive data are mainly stored and efficiently manipulated by some sort of centralized systems that rely on authoritative centers with huge storage capacity and very high throughput. However, these centralized solutions have several inherent drawbacks, including single-point failure and dishonesty of authoritative centers. Once a centralized system is hijacked, the data are at high risk of leakage, tampering, and forgery; thus, these centralized systems cannot be used to enable secure and trustworthy data provenance sharing. Even worse, in many business collaborations, such as supply chain, federated healthcare, or scientific workflow, multiple peers may not fully trust with each other. It is very hard, even not impossible, to elect a trusted center that is acceptable for all peers. A decentralized solution is expected to enable each peer to independently collect and flexibly share provenance views with different users in a secure and trustworthy manner.

A blockchain is a tamper-proof ledger as a chain of blocks. Replicas of the ledger are consistently maintained by peers without mutual trust, enabling trustworthy transactions among multiple peers not relying on a trusted center [7,8]. Some researchers have proposed blockchain-based solutions for trustworthy data sharing by embedding raw data with limited size or their hash values into blockchain transactions [9–11]. Similarly, blockchain-based trustworthy provenance sharing is also feasible [12,13].

Furthermore, a provenance graph could be too large to be directly embedded in a single blockchain transaction, and existing researches of embedding the hash value of a provenance graph into a blockchain transaction or embedding each provenance record in a single blockchain transaction still suffer the drawbacks of low trustworthiness or low feasibility, respectively. First, when only embedding hash values into blockchain transactions for saving storage cost, it is very difficult to maintain the consistency between the hash values on-chain and the provenance graphs off-chain. Once a provenance graph is intentionally redacted for removing sensitive information or maliciously tampered, the evidences on-chain can soon become invalid, and the trustworthiness of data provenance as well as the shared data cannot be verified anymore. Furthermore, provenance graphs may contain various sensitive information that can only be accessed by different requesters with appropriate permissions. Computing provenance views and storing their hash values on-chain after receiving requests leaves much room for provenance owners to falsify the provenance views; therefore, the responsively created provenance views are less trustworthy.

Second, a provenance graph consists of a set of provenance records. Some researchers tried to directly embed each provenance record into a single blockchain transaction. However, when the number of provenance records increase, this method can lead to massive transactions and can soon become infeasible because the cost of spreading and storing these massive transactions on the blockchain network is usually very high, and the throughput of mainstream blockchain platforms is usually too low to timely deal with transactions that could arrive in a very fast speed.

To address the above drawbacks, we proposed the BSTProv, a blockchain-based system for secure and trustworthy decentralized data provenance sharing. The BSTProv system is designed with the following key ideas in mind. First, it embeds encrypted provenance data, but not their hash values, into blockchain transactions to achieve higher trustworthiness at the cost of higher storage cost. To control the storage cost, users could only share trustworthiness-critical provenance graphs with a limited size using the proposed system while dealing with other provenance graphs in traditional approaches. In this sense, the proposed system complements the existing researches by providing users with a new alternative. Second, it deals with a lot of provenance records that are newly ap-

pended onto the provenance graph in a batch mode. In this way, it enables users to adapt the time interval between two consecutive activities of provenance graph sharing and the number of subgraphs to be embedded into blocks to the throughput of underlying blockchain platforms.

With this proposed BSTProv system, a provenance owner can selectively share useful and insensitive provenance subgraphs to specific requesters to prevent sensitive information leakage. Furthermore, a provenance requester can obtain trustworthy provenance subgraphs from blockchain, and can decrypt and compose them into a partially complete but useful provenance graph for further cross-domain provenance tracing and trustworthy provenance-based data trustworthiness enhancement. Note that the proposed system enables different peers to define authorization policies for provenance subgraphs shared on-chain in an autonomous and flexible way.

The remainder of this paper is structured as follows. Section II briefly introduces the preliminaries of blockchain and provenance. Section III describes the related work. Section IV analyzes the fundamental requirements of decentralized provenance management. Section V presents algorithms for provenance graph partitioning and subgraph composition. Section VI presents the high-level architecture of the proposed BSTProv system with emphasis on designing smart contracts and the on- and off-chain storage model. Section VII implements a prototype and highlights the key data structures and algorithms. Section VIII analyses and evaluates the prototype. Finally, Section VIII concludes this paper and envisions future works.

## 2. Preliminaries

### 2.1. Blockchain

Blockchain is initially the set of underpinning techniques of the Bitcoin introduced in 2008 [7], which is the first cryptocurrency system that solved the problem of double-spending and transactions tampering. A blockchain consists of a series of cryptographically linked blocks. Each block has a header and a body that contains a set of transactions. Each block is linked to its previous block by storing the hash of its previous block header.

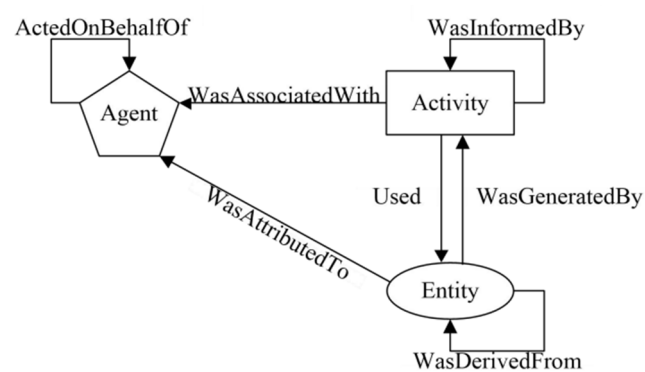
A blockchain is a tamper-proof ledger running on a peer-to-peer network. Each peer maintains an identical replica of the ledger according to a given consensus protocol, such as proof of work, proof of stake, proof of authority, or practical byzantine fault tolerance; thus, multiple peers can achieve consensus without trusted central authorities [8]. In a blockchain network adopting the proof-of-work protocol, a malicious peer who wants to forge a block has to possess more than 50% computational power of the whole network, which is often believed impossible [7]. In this sense, every transaction in a block is immutable and thus, to some extent, trustworthy. Due to these attributes of blockchain, it is possible to build a blockchain-based solution for secure and trustworthy provenance sharing across domains without mutual trust by embedding provenance in blockchain transactions.

Besides decentralized trust and tamper-resistance, newer blockchain systems further provide a decentralized Turing-complete platform to run application codes called smart contract in Ethereum [14] or chain code in Hyperledger [15]. A smart contract in Ethereum is actually a piece of tamper-proof code stored on-chain. It is created and invoked via transactions, and then independently executed by all nodes in the network when specific conditions are met. The execution result of a smart contract obtained on one node is always identical to those obtained on other nodes. A smart contract is not allowed to call codes running off-chain because they may return uncertain results and therefore break the consistency among the ledgers maintained by different nodes. In theory, any decentralized applications with arbitrary complexity can be implemented as smart contracts. In this paper, we use smart contracts to track the authorization policies and high-level dependencies among provenance graphs. Whenever new policies or new graphs are uploaded onto the blockchain, corresponding smart contracts can execute and update the list of authorization policies and the overall dependencies among provenance graphs.

A public blockchain system allows any user to join or exit the network at any time [16]. There are two shortcomings of using a public blockchain for enterprise computing. One shortcoming is the extremely high cost of storing data and executing codes on every node in the public blockchain network. Note that each computational step in the public Ethereum has a cost associated with it being called gas [14]. The initiator of each transaction has to pay the gas price for each step executed in the transaction. The other shortcoming is the low throughput of the public blockchain. Note that a necessary time interval, such as ten minutes in the Bitcoin and fifteen seconds in the Ethereum, is necessary for all nodes to achieve consensus. In order to meet requirements of high throughput and to reduce the cost of storing data and executing codes on a public blockchain, some permissioned blockchains, also named private blockchains or consortium blockchains in different scenarios, were proposed for enterprise computing [15–17]. Users who want to join in a permissioned blockchain have to obtain appropriate permissions from one or more authoritative users. Note that some researchers have employed consortium blockchains to meet high throughput requirements at a relatively low cost [18]. This paper adopts an Ethereum-based consortium blockchain for efficient and cheap provenance storage and tracing.

## 2.2. Provenance

According to W3C, data provenance records the evolutionary history of a data object, and is usually structured into an append-only directed acyclic graph [5]. As shown in Figure 1, a provenance graph contains three types of nodes: Entity, Activity, and Agent. Entity represents intermediate artifacts involved in producing a data object. Activity represents processes that manipulated entities. Agent represents persons or organizations that controlled activities. An edge  $\langle t, h \rangle$  is a directed dependency among nodes and is actually a provenance record with a timestamp that indicates an event from the past. For example, an activity  $a_1$  can use a data object  $o_1$  as the input, a new data object  $o_2$  can be generated by an activity  $a_2$ , and one agent  $g_1$  can act on behalf of another agent  $g_2$ . An edge can be created and appended into a provenance graph whenever a concerned event happened. Two edges can be connected together when the head of one edge  $p = \langle t_1, h_1 \rangle$  is equal to the tail of another edge  $q = \langle t_2, h_2 \rangle$ , i.e.,  $h_2 == t_1$ . In this case, the event modeled by  $p$  happened later than the event modeled by  $q$ . We can say that  $p$  is a downstream event of  $q$  or that  $q$  is an upstream event of  $p$ .



**Figure 1.** Core structure of a provenance data model. Reprinted with permission from Ref. [14]. 2014, Wood, G.

Along the direction of edges in a provenance graph, one can trace and scrutinize the historical nodes and events that directly or indirectly influenced a node. This process is called provenance tracing, which is the main usage of a provenance graph. Provenance tracing is the prerequisite for data trustworthiness verification, wrong data attribution, and accountability. We use the term traceability to indicate the extent to which a provenance graph facilitates provenance tracing. One objective of the proposed system is to ensure

traceability of a cross-domain provenance graph that is composed from some subgraphs shared on-chain.

### 3. Related Work

The locations where provenance is collected and stored lead to different techniques for provenance sharing and therefore different challenges. In a centralized setting where provenance is collected and stored in an authoritative center, the main concerns are security and traceability because the trustworthiness of the shared provenance graphs is tacitly ensured by the authoritative center. Most existing researches focused on ensuring provenance security by techniques of encryption [19,20], sanitization [21], and access control [22,23]. Some researches further considered the issue of achieving both security and traceability [24]. However, the trustworthiness of the shared provenance is doubtful because the centralized structure has downsides of inevitable single-point failure and center dishonesty. The malicious center can fabricate or falsify critical data and attribute its crime to data providers. Even worse, it is hard to establish an authoritative center in collaborations across organizations, such as supply chain and federated healthcare.

With the advent of blockchains, many researchers explored the possibility of blockchain-based data sharing among peers without mutual trust. Liu et al. proposed a privacy-preserving medical data sharing system based on blockchain [25], which stores the medical information on a cloud and its hash value on a consortium blockchain. Wei et al. built a blockchain-based data integrity protection mechanism, which enables reliable data storage, monitoring, and verification in cloud [26]. Xia et al. designed the MeDShare system, which uses smart contracts and access control mechanisms to not only trace and control data effectively, but also to minimize the risk of private data leakage [27]. RiFi et al. stored data of Internet of Things by IPFS and controlled access by smart contracts [28], which can protect privacy and sensitive data. Blockchain has shown great potential in addressing privacy and security vulnerabilities in Internet of Things (IoT) and many researches have been conducted concerning security challenges of smart contracts and the performance evaluation [29–31]. Previous researches have shown that blockchain-based solutions are feasible for secure and trustworthy data sharing if appropriate storage patterns, smart contracts, and encryption technologies are well integrated.

Inspired by researches on blockchain-based data sharing, some researchers also explored the possibility of blockchain-based data provenance sharing in distributed environments. Xu et al. proposed a provenance-provided data sharing model based on blockchain [10], which has features of transparent authentication, privacy control, and auditable provenance. Ramachandran et al. use smart contracts and an OPM model to record, verify, and manage data provenance [32]. Chen et al. designed a block structure for storing and retrieving data provenance [33], which certificates the primitive data on-chain and saves them off-chain. Liang et al. proposed a ProvChain framework for collecting and verifying cloud data provenance [12], which embeds the provenance of cloud files into blockchain transactions. Jagadeesh et al. proposed an extensible framework based on blockchain for capturing, storing, exploring, and analyzing software provenance [34]. Fernando et al. developed a system called SciBlock that provides tamper-proof and undeniable storage for scientific workflow provenances [13]. Ruan et al. proposed a fine-grained, secure, and efficient provenance system called LineageChain, which motivates the need for adding native provenance support to blockchain [35].

Besides the naïve and usually infeasible method of embedding the whole provenance graph directly into a single blockchain transaction, existing blockchain-based provenance sharing researches mainly fell into two categories. One is to embed a single provenance record in a blockchain transaction. The other is to embed the hash value of the whole provenance graph in a blockchain transaction. The researches of the first category are usually infeasible when a large number of provenance records emerge at a very fast speed. First, a large number of transactions should be created and spread on the blockchain network in a short time period, which cannot be facilitated by existing blockchain systems.



Second, what is used in provenance tracing is actually a provenance graph consisting of multiple interconnected provenance records. Most existing researches did not address the issue of efficiently integrating provenance records scattered in different blocks. Third, managing access rights of a large number of provenance records is also very cumbersome and error-prone.

The researches of the second category have two drawbacks of low trustworthiness and inflexible authorization. First, they suffer the low trustworthiness of modified provenance file off-chain because any modifications to a provenance graph off-chain cannot be validated by the corresponding hash value stored on-chain. Furthermore, constructing provenance views and their corresponding hash values according to requests leaves much room for provenance owner cheating; thus, provenance views constructed in this way are less trustworthy. Second, they suffer the inflexible authorization because a provenance graph cannot be selectively shared and validated because only the hash value of the whole provenance graph was on-chain. Users who only need a subset of the provenance graph can acquire unnecessary or even sensitive information when the provenance was shared as a whole. In addition, one still faces the challenges of constructing, processing, and maintaining massive provenance views, which may contain only subtle differences with each other in terms of sensitive information, and faces the challenges of distributing massive decryption keys to specific users.

The merits and demerits of existing researches are summarized in Table 1. Existing centralized solutions focused on security and traceability; thus, they have the merits of high security, efficiency, and traceability, but the demerit of low trustworthiness. Existing blockchain-based solutions focused on trustworthiness and security, but not traceability; therefore, they have the merits of high security, but the demerits of low trustworthiness, high cost and complexity of managing a large number of provenance records and their access policies, and low traceability.

**Table 1.** A comparison of schemes.

Schemes	Research Focus	Methods	Merits	Demerits
Centralized solutions	Security and traceability	Data encryption Data sanitization access control	High security High dependability High traceability	Low trustworthiness High risk of data forgery Disputes of responsibility
Blockchain-based solutions	Security and trustworthiness	Smart contract  Data encryption	High security  Trustworthiness	Low traceability when storing provenance records separately High cost of managing massive provenance records High complexity of managing massive access policies Low dependability when only storing hashes on-chain

#### 4. Blockchain-Based Decentralized Provenance Sharing

In a decentralized business collaboration across multiple organizations, such as supply chain, scientific workflow, or federated healthcare, peers may not fully trust each other. Each peer may collect provenance records by itself and store them in a local database. Along with the system running, each peer can capture a lot of provenance records and structure them into a local provenance graph and then share them with other peers when necessary. Some shared local provenance graphs can be composed into a cross-domain provenance graph for global provenance tracing.

A local provenance graph may include various sensitive or redundant information for a specific requester to consume. If a peer shared its whole local provenance graph with other peers without any modifications, they may not only suffer the high cost of transferring and processing a large provenance graph, but also the high risk of sensitive information leakage. In practice, each peer needs to share only necessary subgraphs to

specific peers while ensuring that they are secure and trustworthy. If a data requester wants to trace a given entity back to its origins, he/she can request provenance subgraphs shared by multiple peers, verify their trustworthiness, and then compose them into a trustworthy provenance graph. In general, data provenance sharing in decentralized settings mainly has three objectives: security for preventing unauthorized users from reading and accessing sensitive information, traceability for retaining enough useful information in the cross-domain provenance graph composed from shared subgraphs, and trustworthiness for keeping the cross-domain provenance graph authentic.

As analyzed above, several fundamental requirements of decentralized provenance sharing are identified as follows:

1. What a peer shares each time is not a single provenance record but a provenance subgraph consisting of a bundle of interconnected provenance records.
2. Each peer should construct appropriate subgraphs, encrypt them, and define access control policies for securely and flexibly sharing them with different users.
3. Provenance subgraphs shared by different peers should be efficiently composable with each other for cross-domain provenance tracing.
4. Provenance subgraphs shared by a peer should be trustworthy and dependable even when the provenance graph locally maintained by the peer is damaged or tampered accidentally or intentionally.

This paper proposes a blockchain-based secure and trustworthy data provenance (BSTProv) sharing system. With the BSTProv system, a provenance owner first works locally to partition a provenance graph into multiple subgraphs, to encrypt each subgraph using a unique symmetrical encryption key, to embed each encrypted subgraph instead of its hash value into a blockchain transaction, and to submit the transactions onto the blockchain network. Second, the blockchain network spreads the transactions. Third, the miner elected according to some consensus protocol, such as PoS, packages these transactions into a block and spreads the block across the network. Fourth, provenance owners can independently define and submit policies onto the blockchain network to authorize a requester to access a set of subgraphs. Meanwhile, provenance requesters can independently define and submit requests onto the blockchain network to request a set of subgraphs. Note that provenance owners and provenance requesters can work independently and simultaneously. At last, each provenance requester can retrieve his/her permissions and download corresponding subgraphs from the blockchain. He/she can then compose these subgraphs obtained from the blockchain into a cross-domain provenance graph.

The following sections introduce in detail the provenance partitioning and composition mechanisms, the overall architecture, and the implementation of the BSTProv system, respectively.

## 5. Provenance Partitioning and Composition

In this paper, a local provenance graph maintained by a peer should be partitioned into multiple subgraphs, which are then encrypted and uploaded onto the blockchain. Different users can then obtain a subset of provenance subgraphs from the blockchain and compose them into a partially complete provenance graph for cross-domain provenance tracing. This section introduces a breadth-first search-based provenance graph partitioning algorithm and a hash-node based provenance graph expansion algorithm for re-composition of inner-domain provenance subgraphs or intra-domain provenance graphs.

### 5.1. Partitioning

Graph partitioning is generally the first step of distributed graph computing tasks. The objective is to find balanced partitions of a graph while minimizing the number of edge cut [36,37]. In this paper, a large provenance graph should be partitioned into multiple small subgraphs so that they can be embedded into blockchain transactions with limited capacity and flexibly authorized to different users. Thus, each provenance subgraph should be small enough and should only include sensitive elements that can be

authorized to a user together. This represents a graph partitioning problem with security constraints. A provenance graph may contain various sensitive information in its nodes, edges, or even indirect dependencies among two nodes connected via a path. A security constraint to provenance graph partitioning could tell that each sensitive node, edge, or endpoint of an indirect dependency in a provenance graph should be encapsulated in an independent subgraph so that one can authorize specific users to access a sensitive element by authorizing them to access the subgraph including it.

Graph partitioning is a combinatorial optimization problem that has been exhaustively studied [36]. Existing graph partitioning algorithms cannot be straightforwardly used for provenance graph partitioning with security constraints. A directed edge or arc in a provenance graph usually indicates an informal casual dependency with an arc tail as the effect and an arc head as the cause. In order to respect the traceability of provenance subgraphs, this paper introduces a breadth-first search-based provenance graph partitioning algorithm. It creates a subgraph by grouping a node or an effect with some of its recent causes that are upstream nodes within given hops from the node.

The proposed algorithm takes the provenance graph  $G$ , the maximum length of paths in a subgraph  $N$ , and the set of sensitive elements  $S$  as the input, and produces a set of subgraphs  $G_i, i = 1, \dots, |S|$  as the output. After starting a breadth-first search from a node with indegree zero, any newly reached nodes, as well as the edges reaching them, can be added into a subgraph  $G_i$ , unless a path whose length exceeds the given value  $N$  or the new node or the new edge is sensitive. In this way, the algorithm ensures that each  $G_i$  includes only one sensitive element in  $S$ . One can define authorization policies to protect a subgraph from being accessed by unauthorized users. When a subgraph  $G_i$  is created, 1 should be subtracted from the indegree of each node when it is reached from  $G_i$  via an untouched edge. Then, the next subgraph  $G_{i+1}$  can be computed by starting another breadth-first search from a node with indegree zero. The proposed algorithm stops when all nodes in  $G$  are added into some subgraphs.

Figure 2 shows a partial provenance graph of an entity EmailC. In Figure 2, the edge  $\langle \text{EmailC}, \text{Send2} \rangle$  means that the entity EmailC was generated by the activity Send2. The edge  $\langle \text{addContent}, \text{file} \rangle$  means that the activity addContent manipulated the file. Assume that the entity EmailA is sensitive to the provenance requester A, that the entity file is sensitive to the provenance requester B, and that the maximum length of paths in a subgraph is set as 2. Executing the proposed partitioning algorithm starting from the entity EmailC can produce three provenance subgraphs S1, S2, and S3, as shown in Figure 2. The data owner can share only S1 and S2 to the data requester A to avoid the leakage of the sensitive EmailA in S1.

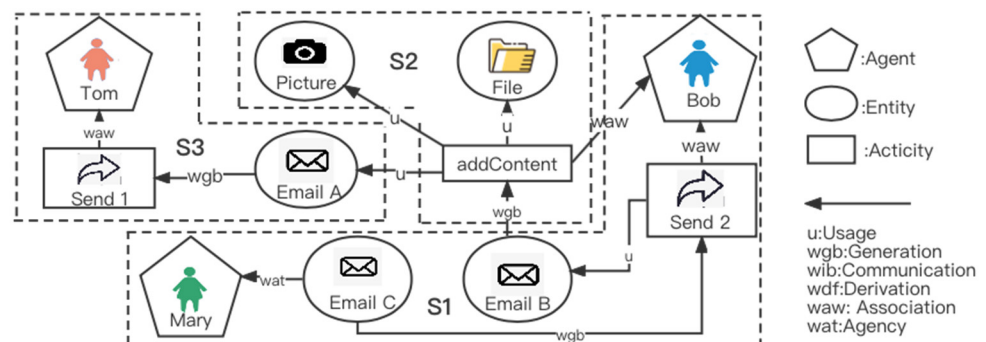


Figure 2. A provenance graph of email.

Note that the proposed algorithm is not necessarily optimal. Many other graph partitioning algorithms can be applied. Exploring new provenance graph partitioning algorithms is important in and of itself, and out of the scope of this paper. For example, a subgraph was shared to a user as a whole may become partially available to another user when authorization policies change in the future. However, our approach does not



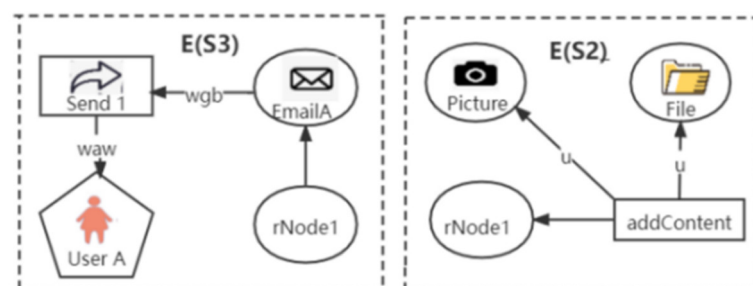
divide a submitted subgraph to keep it trustworthy. Enabling adaptable and trustworthy partitioning to a submitted subgraph is very interesting and is left as our future work.

## 5.2. Composition

Provenance graphs maintained by different peers in a business collaboration usually share some nodes which are the data objects transferred across different domains; thus, they can be composed into a cross-domain provenance graph for global provenance tracing by merging these common nodes. According to the above algorithm, a provenance graph can be partitioned into multiple non-overlapping subgraphs connected via only one or more cut edges, for example  $\langle \text{addContent}, \text{EmailA} \rangle$  between S2 and S3. Simply removing the cut edges can not only lose the information on these edges, but also totally disjoint subgraphs that cannot be composed again. To this end, this paper proposed a hash-node-based provenance graph expansion algorithm for both inner-domain subgraph composition and intra-domain graph composition. It expands a provenance graph by introducing a few dumb nodes and incident edges so that the expanded provenance graph can be composed with other expanded provenance graphs via common dumb nodes. Each dumb node is actually the hash value of either a common node of two graphs or a cut edge connecting two subgraphs.

In order to compose two subgraphs without common nodes but connected via a cut edge, the proposed algorithm first removes the cut edge; adds a dumb node and a dumb edge, connecting the dumb node and one of two endpoints of the cut edge in each subgraph, respectively; and finally attaches the edge information, if it exists, to the dumb edge incident with the tail of the cut edge. Here, the dumb node is actually the hash value of the cut edge. As a result, each subgraph is equipped with an array of input dumb nodes and an array of output dumb nodes which are composable with other downstream or upstream subgraphs, respectively. Two expanded subgraphs can then be composed together by merging their common dumb nodes and incident dumb edges into a cut edge. In this way, the algorithm enables the inner-domain subgraph composition without introducing significant storage cost.

For example, Figure 3 shows the expansions of subgraphs S2 and S3 in Figure 2. The hash value of the cut edge  $\langle \text{addContent}, \text{EmailA} \rangle$  is added as the dumb node  $rNode1$ , and its two incident dumb edges ( $\langle \text{addContent}, rNode1 \rangle$  and  $\langle rNode1, \text{EmailA} \rangle$ ) are also added. Edge information on  $\langle \text{addContent}, \text{EmailA} \rangle$  is attached to the edge  $\langle \text{addContent}, rNode1 \rangle$ . The dumb node  $rNode1$  and two incident dumb edges ( $\langle \text{addContent}, rNode1 \rangle$  and  $\langle rNode1, \text{EmailA} \rangle$ ) in Figure 3 can be merged into  $\langle \text{addContent}, \text{EmailA} \rangle$  later.



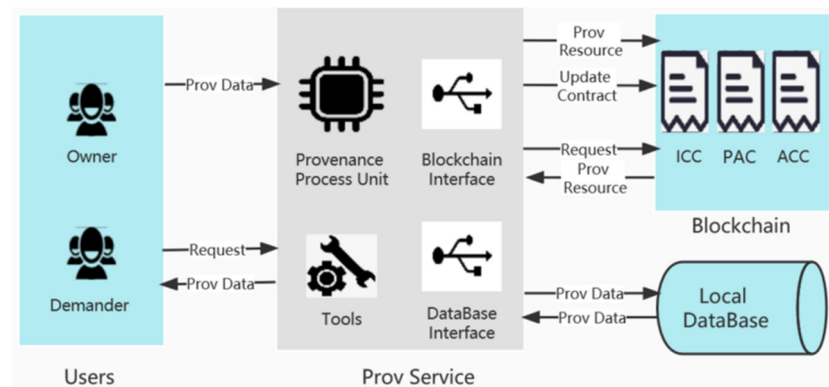
**Figure 3.** Expansion of provenance subgraphs of Figure 2.

In order to compose two provenance graphs sharing a common node, the proposed algorithm introduces into the two provenance graphs, i.e., a dumb node that is actually the hash value of the common node, and an incident dumb edge that connects the dumb node and the common node. Specifically, in a given provenance graph, it introduces dumb nodes for nodes with indegree zero to form the array of *nextIndex*, and dumb nodes for nodes with outdegree zero to form the array *priorIndex*, respectively. Both *priorIndex* and *nextIndex* can then be stored in a smart contract and serve as its cross-domain hooks to other provenance graphs maintained by different peers.

By matching common dumb nodes, the proposed algorithm enables efficient and trustworthy cross-domain composition of provenance graphs and provenance tracing even when some peers did not share some of their subgraphs. This will be discussed in detail in the next section. By using hash values as hooks, the proposed algorithm avoids the unnecessary leakage of sensitive information in the original common nodes. If a peer *A* sends a data object *D* to another peer *B* in a business collaboration, the transferred data object *D* is certainly available to both its sender *A* and its receiver *B*. Both *A* and *B* can compute the hash value of *D* independently. However, other peers involved in the business collaboration are informed that a message is passed from *A* to *B* while they do not know the content of the data object *D*.

## 6. Architecture

As shown in Figure 4, the BSTProv system consists of four types of modules: users, prov service, local database, and consortium blockchain. Users are the people or organizations that deal with business tasks by some centralized systems which collect and share various business data objects and provenance with other users for various purposes. Users served by a centralized business system belong to an administrative domain and are abstracted as a peer here; thus, we view users and peers as two interchangeable terms in the rest of this paper. Multiple peers in a business collaboration may exist. Each peer needs to maintain a trusted prov service and a trusted blockchain node so that it can process its own provenance independently and efficiently and join the consortium blockchain network for trustworthy provenance sharing among peers.



**Figure 4.** An architecture of the BSTProv system.

A prov service usually runs in a trusted zone for a specific peer and consists of a provenance process unit, a blockchain interface, a database interface, and a set of tools. The provenance process unit is used to partition a local provenance graph into multiple small subgraphs and compose multiple subgraphs obtained from the blockchain into a cross-domain provenance view. The blockchain interface and the local database interface is used to interact with the underlying blockchain and the local database, respectively. A set of tools is used to manage local asymmetric encryption key pairs, and to encrypt and decrypt provenance subgraphs.

The consortium blockchain stores provenance subgraphs, as well as the overall dependency structure about how cross-domain provenance graphs can be composed with each other. It selectively shares provenance subgraphs with appropriately designed smart contracts.

To ensure efficient data storage and query, the local database stores the local provenance graph collected and maintained by each user, provenance subgraphs (either the partitions of the local provenance graph or those shared by other users), the information of the transaction and block where a subgraph is embedded in, and access control policies defined for requesters.

In the process of provenance sharing, a provenance owner first sends a provenance graph to the trusted prov service. Then, the prov service partitions the provenance graph

into multiple subgraphs and computes cross-domain dependencies, i.e., the *priorIndex* array and the *nextIndex* array of the provenance graph, according to the proposed algorithms for provenance graph partitioning and expansion. Thirdly, the prov service sends the *priorIndex* array and the *nextIndex* array onto a predefined smart contract running on the consortium blockchain. The smart contract parses the transaction and links the current provenance graph to existing ones. Finally, the prov service sends the encrypted subgraphs onto the blockchain one by one and stores the transaction receipts in the local database.

In the process of provenance requesting, a provenance requester can first send a provenance request to the trusted prov service. The prov service then retrieves the user's permissions in a smart contract running on the consortium blockchain, and further downloads the shared provenance subgraphs according to the retrieved permissions. It then decrypts and composes the subgraphs and stores them in the user's local database. Note that smart contracts and on-and-off storage patterns are two important design issues that we be introduced in detail in the rest of this section.

### 6.1. Smart Contract

Existing researches have built consortium blockchains on top of the Ethereum in different applications [38,39]; therefore, we adopted the Ethereum as the underlying platform to build our system and used smart contracts to implement the on-chain computations. A smart contract has a unique address and a corresponding storage tree, which is a part of the global state of the Ethereum. A data-storing transaction can be sent to a smart contract and the appended data objects can be extracted and stored in the storage tree.

In the BSTProv system, we designed three smart contracts to manage authorization policies and high-level dependencies among multiple provenance graphs from different domains for decentralized authorization and cross-domain provenance composition, including the identity controller contract (ICC), the provenance association contract (PAC), and the authority controller contract (ACC).

The ICC records the identifications of users and registers new users by voting. The ICC consists of the identity storage smart contract (ISSC) and the identity register voting contract (IRVC). The ISSC stores the unique identifications and the corresponding public keys of registered users. The IRVC creates a vote request whenever a new user registration request arrives. When receiving more than half the number of votes in a given time period, the ICC grants the user's registration request and stores the user's address and its public key on-chain.

The PAC manages the overall dependency structure among provenance graphs maintained by different peers, which is key to facilitating cross-domain provenance tracing. The PAC consists of the provenance graph indexing contract (PGIC), the provenance graph associating contract (PGAC), and the provenance upload smart contract (PUSC). The PUSC parses the transactions with the embedded index information of a provenance graph and then invokes the smart contracts PGIC and PGAC. The PGIC stores the index information of a provenance graph, including the graph number, the owner, and the arrays of *priorIndex* and *nextIndex*. The two arrays are stored in the PGIC for keeping cross-domain dependencies among provenance graphs. It uses the hash value of a node instead of the raw node itself to avoid the leakage of sensitive information. The PGAC stores the overall dependency structure of provenance graphs for efficient cross-domain graph composition.

The ACC enables a provenance owner to define access policies for requesters and accepts provenance requests issued by data requesters. The ACC consists of the provenance request smart contract (PRSC) and the access control smart contract (ACSC). The PRSC stores provenance requests issued by requesters. The ACSC stores access policies published by provenance owners. A provenance owner can define who can access which subgraphs of a provenance graph as an access policy. Note that an access policy only refers to the public key of a provenance requester for preserving privacy.

### 6.2. Storage On-Chain and Off-Chain

Properly off-chaining data and computation can save the high cost of storing a large amount of data and of conducting complex computations on blockchain [40]. By carefully storing provenance-related information on-chain or off-chain, the BSTProv system guarantees the security and trustworthiness of the shared provenance while limiting the operation cost for provenance-related storage and computation. On-chain storage will be discussed in detail in the next section along with the on-chain computations, i.e., smart contracts. The rest of this section elaborates off-chain storage.

In the BSTProv system, each peer maintains a local database that not only stores the local provenance graph, corresponding provenance subgraphs, access control policies, provenance requests that are sent onto the blockchain, the subgraphs, and transaction receipts that are obtained from the blockchain, but also all other information that is useful for accelerating related computation. Three key tables in the local database are shown in Figure 5. The provenance\_data table is used to store the local provenance graph and cross-domain dependencies to be uploaded onto the blockchain, i.e., the arrays of priorIndex and nextIndex. The blockchain\_storage table is used to store a provenance subgraph, its symmetric encryption key, and its location on the blockchain. The permission\_allow table is used to store the set of permissions defined by provenance owners for flexibly sharing different combinations of provenance subgraphs to different provenance requesters.

provenance_data	blockchain_storage	permission_allow
pId proveFile remark datetime	dId disProvFile flag location key datetime	aId demander permission datetime

Figure 5. Relations in the local database of a provenance owner.

In the process of data provenance sharing, the fields in these tables can change dynamically. A provenance owner first uploads a provenance graph and the prov service then stores it into the provenance\_data table. The prov service then partitions the provenance graph into several subgraphs, encrypts them symmetrically, and stores them in the blockchain\_storage table. Fields location and flag are left empty temporarily.

The prov service then sends the encrypted provenance subgraphs onto the blockchain. When the provenance subgraph is successfully stored on the blockchain, the prov service sets the flag field as true and the location field as the location of the provenance subgraph on the blockchain in the blockchain\_storage table. If a provenance subgraph fails to be stored on the blockchain, the prov service can set the flag field as false and the location field as empty in the blockchain\_storage table.

After successfully sending provenance subgraphs onto the blockchain, the provenance owner can define permissions for different data requesters according to the related information in the local database. The prov service can then store these permissions into an on-chain smart contract and into the local permission\_allow table, respectively. Note that users without permissions cannot decrypt the encrypted provenance subgraphs embedded in blockchain transactions therefore the security of provenance on-chain is ensured.

### 7. Implementation

We built a prototype of the BSTProv system on the Ethereum platform, using solidity 0.4.24, node.js and truffle framework, and used the Geth v1.10.13 as the Ethereum client to set up an Ethereum test network. The rest of this section elaborates how main functions of the BSTProv system are implemented.

### 7.1. Identity Registration

Different peers that want to exchange trustworthiness-critical data objects in a decentralized business collaboration can join the BSTProv system for trustworthy provenance sharing. A registration request is granted or denied by the ICC smart contract that supports majority voting [41]. No third parties can intervene in the process or the voting because different peers may vote via their local node, and the result of voting is stored and counted by the public and immutable codes in the smart contract ICC.

Specifically, a user first sets a private string and uses it to generate an asymmetric key pair. Then, he/she logs into the prov service through the public key and sends a registration request to the ICC. The ICC accepts the request and initiates a voting proposal for it. Each authenticated user can evaluate the registration request and make a decision for or against it independently. Finally, the ICC can collect and verify each vote in the given voting period. When the voting period ends, the ICC stores the user's address and the public key if the total number of votes is greater than the required number of votes. The ICC is shown in Algorithm 1 as follows.

---

#### Algorithm 1: ICC.

---

```

Procedure Registration (user, pk, request)
  issc = web3.eth.Contract (ISSC_ADDRESS)
  irvc = web3.eth.Contract (IRVC_ADDRESS)
  allUsers[] = issc.methods.getUser()
  if (allUsers.contains (user, pk)) // user existed
    return;
  irvc.methods.vote(user, pk, request) // notify authenticated users to vote
  sleep (VOTING_TIME); // waiting for users voting
  proposal = irvc.methods.getvotes (user, pk)
  if (proposal.winnerVotes > proposal.totalVotes*0.5) then
    issc.methods.register (user, pk);
  return;
End procedure

```

---

### 7.2. Provenance Uploading

The process of provenance uploading includes a series of operations which are as follows.

Step 1: A provenance owner first sends his/her provenance graph to a trusted prov service. The prov service then computes the hash values of input and output nodes in the provenance graph and stores them in the array *priorIndex* and array *nextIndex*, respectively.

Step 2: The prov service partitions the provenance graph with the proposed partitioning algorithm. The provenance owner can declare appropriate parameters to adjust the partitioning results, including a set of sensitive elements and the maximum length of paths in a subgraph. The prov service can randomly generate a unique encryption key for each provenance subgraph and encrypt it symmetrically.

Step 3: The prov service then sends a transaction with the cross-domain dependencies of the provenance graph to the smart contract PUSC. The PUSC parses the transactions and calls the smart contract PGIC to extract and store the index information of the provenance graph, including the graph number, the address of its owner, and its hooks to other graphs (two arrays of *priorIndex* and *nextIndex*). Then, the PUSC invokes the smart contract PGAC to link the current provenance graph to other existing ones.

Step 4: The prov service sends transactions to upload encrypted provenance subgraphs onto the blockchain one by one. When the blockchain state is globally updated, the prov service can retrieve transaction receipts to extract the locations of the provenance subgraphs and store them together with corresponding secret keys into the local database of provenance owners. Note that sufficient gas price should be set to each transaction in order to upload the embedded provenance subgraph successfully.

Figure 6 shows the state of the smart contracts PGIC and PGAC after a series of provenance uploading requests were processed. The PGIC stored the index information of

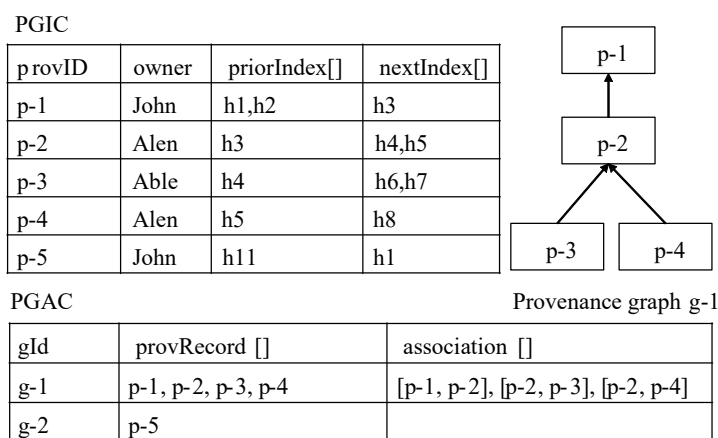


provenance graphs that are numbered p-1 and are shared by the provenance owners John, Alen, and Able, respectively. Note that the name of a provenance owner can be replaced with his/her public key or blockchain address for preserving privacy in practice. Whenever the index information of a provenance graph is stored in the PGIC, the PGAC is invoked to link the newly added provenance graph to existing provenance graphs by traversing and matching their priorIndex and nextIndex, or to create a new cross-domain provenance graph when necessary. For example, g-1 is created when the graph p-1 is uploaded. Then, p-2 is added into g1 because of the prior index of p-2 is h3 that is identical to the next index of p-1. Furthermore, g-2 is created when the graph p-5 that cannot be linked to any existing graphs is uploaded. The smart contract PUSC is shown in Algorithm 2.

**Algorithm 2:** PUSC.

```

Procedure UploadProvIndex (owner, graphNum, address, priorIndex [], nextIndex [])
  pgic = web3.eth.Contract (PGIC_ADDRESS)
  pgac = web3.eth.Contract (PGAC_ADDRESS)
  graphNums [] = pgic.methods.getgraphNum()
  if (graphNums.contains (graphNum)) //provenance graph existed
    return;
  //save overall dependencies among provenance graph
  pgic.methods.saveProvIndex (graphNum, owner, priorIndex, nextIndex)
  pgac.methods.updateAssociation (graphNum, owner, priorIndex, nextIndex)
End
    
```



**Figure 6.** Snapshots of the storage of the smart contracts PGIC and PGAC.

**7.3. Provenance Authorizing**

After uploaded onto blockchain, provenance subgraphs can be authorized to different peers. Provenance owners can define appropriate permissions to grant or deny provenance requests of different requesters. The specific process of provenance authorizing is as follows.

Step 1: A provenance requester sends a provenance request to the smart contract ACC.

Step 2: The ACC verifies the identity of the requester and broadcasts the request to all the provenance owners in the blockchain network.

Step 3: A provenance owner analyzes the provenance request and identifies the required provenance subgraphs that can be shared to the requester, and then defines and stores appropriate permissions in both the ACC on the blockchain and the local database. In practice, a provenance owner can also define permissions in advance to grant specific requesters to access some provenance subgraphs before the requesters submit requests.

Figure 7 shows the state of the smart contract ACC after a series of provenance requests were processed and authorized. A permission in ACC includes four fields: the provenance owner, the provenance requester, the pairs of the location and key of shared targets, and the number of the provenance graph. When a provenance owner John receives the request issued by Mary, he first selects some provenance subgraphs required by Mary in the local

database and then defines a permission with pairs of locations and keys [location, key] in the *authAssign* field in the local database. He uses the public key registered by Mary in ISSC to encrypt the *authAssign* field and then submits the permission with the encrypted *authAssign* field into the smart contract ACC so that this permission can only be decrypted by Mary.

PRSC

userID	require
John	Express Number
Mary	Milk produced on Aug.1,2021

ACSC

owner	demander	authAssign	provID
John	Mary	[loaction1, key1], [loaction2, key2],...	p-1
Alen	Mary	[loaction3, key3], [loaction4, key4],...	p-2
Able	Mary	[loaction6, key6], [loaction7, key7],...	p-3
Alen	John	[loaction9, key9]	p-4

Figure 7. Snapshots of the storage of the smart contract ACC.

7.4. Provenance Retrieval

After provenance owners upload necessary permissions to the smart contract ACC, each requester can then retrieve appropriate permissions from ACC. The specific process of provenance retrieval and composition is as follows:

Step 1: A provenance requester retrieves his/her permissions from the smart contract ACC.

Step 2: The ACC verifies the identity of the requester and returns permissions issued to the requester by different provenance owners.

Step 3: The prov service decrypts the permissions using the private key of the requester to obtain locations and symmetrical decryption keys of the encrypted provenance subgraphs on-chain. Then, it retrieves the encrypted provenance subgraphs from the blockchain and decrypts them.

Step 4: The prov service retrieves the overall dependency structure on how cross-domain provenance graphs can be composed together from the smart contract PAC. Guided by the high-level blueprint, it can compose the acquired provenance subgraphs into a cross-domain provenance graph even when some requested provenance subgraphs are not available. Thus, the proposed system enables the cross-domain provenance tracing. Figure 8 shows the sequence diagram of provenance retrieval.

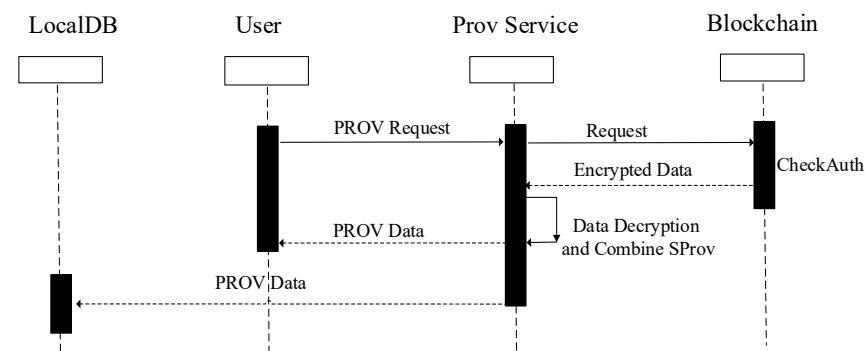


Figure 8. Sequence diagram of provenance retrieval.

For example, Mary can find the overall dependency structure of the cross-domain provenance graph  $g-1$  in PGAC, as shown in Figure 6. Then, she can acquire the permissions related to  $p-1$ ,  $p-2$ , and  $p-3$  from ACSC; retrieve corresponding subgraphs of  $p-1$ ,  $p-2$ , and  $p-3$  at given locations; and compose them into a global provenance graph. If the graph  $p-2$  is not granted by Alen and, therefore, is not available to Mary, Mary still knows how to compose subgraphs of  $p-1$  and those of  $p-3$  together for partially global provenance tracing, according to the overall structure of  $g-1$  in Figure 6.

## 8. Evaluation and Analysis

We evaluated the proposed BSTProv system in terms of its trustworthiness, security, traceability, and storage cost by conducting experiments using the NCFS dataset [42], a dataset from Indiana University simulating the real scientific workflow. We also compared the proposed BSTProv system with existing solutions to show its advantages.

We set up an Ethereum test network with three virtual hosts configured with 16G RAM Intel Core i7 CPU 3.5 GHz running windows. By configuring the Ethereum test network locally, we avoided possible network failures during experiments. We used the PoS consensus mechanism to reduce the performance overhead of block mining. In Ethereum, the default block-out time was 12–14 s and the default block size was 15 million gases. However, these default configurations of Ethereum cannot meet the requirements of dealing with the NCFS dataset. We defined a genesis block that sets the block-out time as 5 s and the block size as 40 million gases to avoid possible transaction delay in our experiments.

### 8.1. Trustworthiness

Trustworthiness of the shared provenance is a key concern of decentralized provenance sharing. The BSTProv system enhances provenance trustworthiness in three ways. First, it stores encrypted provenance subgraphs on blockchain, which keeps them immutable. Once any intentional or accidental falsifications to a provenance subgraph are identified, the peer who uploaded the erroneous provenance subgraph can be easily identified and punished according to predefined business contracts.

Second, the BSTProv system partitions the raw provenance graph into multiple subgraphs, and then stores encrypted subgraphs on blockchain instead of their hash values. In this way, the proposed BSTProv system ensures that provenance graphs are available and trustworthy even when some trustless peers failed or their local databases were tampered or damaged.

Third, a requester may only request a subgraph of a provenance graph to fulfill their business tasks. In the BSTProv system, a provenance owner can only define access policies after uploading provenance graphs onto the blockchain; however, it cannot fabricate a provenance view and store its hash value onto the blockchain chain to cheat requesters. In this way, it enhances the trustworthiness of provenance by preventing malicious provenance owners from falsifying provenance views.

Note that the BSTProv system promises high trustworthiness of shared provenance subgraphs with a relatively high cost of storage. In practice, each peer can selectively share some of critical subgraphs of a provenance graph and leave other uncritical subgraphs shared in a low-cost solution, such as a traditional centralized solution. In this way, the proposed solution complements the existing approaches by providing a method for sharing critical provenance in a trustworthy manner.

### 8.2. Security

Security is an important concern in decentralized provenance sharing. The BSTProv system implements functional security by three means. First, it symmetrically encrypts subgraphs and embeds them onto the blockchain so that requesters without permissions cannot decrypt them. Second, it asymmetrically encrypts permissions using the public key of requesters registered in the smart contract ICC so that only authorized requesters

can obtain the symmetric keys for decrypting the encrypted subgraphs. Third, it honors the least privilege principle by allowing a provenance owner to define access policies, to authorize a specific requester, and to access a specific set of provenance subgraphs, therefore preventing a requester from acquiring unnecessary and even sensitive information.

In practice, each peer joining the provenance sharing network is not fully trustworthy. As discussed in Section 8.1, malicious provenance owners can be punished once any intentional or accidental falsifications to a provenance subgraph are identified. However, the BSTProv system still cannot prevent authorized but malicious requesters to leak sensitive subgraphs legitimately obtained from the blockchain. Even worse, multiple malicious requesters can collude to reconstruct the provenance graph from subgraphs obtained from the blockchain by simply sharing their private keys with each other. Further research is expected to address these issues in the future.

In addition, privacy is another important concern in general data sharing. However, in the scenario of provenance sharing, a more important requirement is to clarify the identities of users who are responsible for some activities or entities. This paper focuses on sharing critical provenance information in a secure and trustworthy manner, but does not pay extra attention to the issue of privacy protection. In fact, the proposed system still provides the basic capability for privacy protection because the underlying blockchain network uses a public key as an anonymized user identity.

### 8.3. Traceability

Provenance traceability is the extent to which a provenance graph facilitates provenance tracing. The proposed system only shares a set of subgraphs to a specific requester. It maintains the overall dependency structure among subgraphs so that a requester can compose subgraphs obtained from the blockchain into a global provenance graph for later provenance tracing. Note that by introducing appropriate dumb nodes hooking different provenance graphs, the proposed system can guarantee the composability of provenance subgraphs, even with some provenance subgraphs missing.

Figure 9 simulates a portion of a cross-domain provenance graph. The provenance graphs  $g_1$ ,  $g_2$ , and  $g_3$  were uploaded by three provenance owners. Each round rectangle in each provenance graph represents a partitioned subgraph. For example,  $g_1$  was partitioned into  $A_1$ ,  $A_2$ , and  $A_3$ . Suppose that a user obtained subgraphs  $A_1$ ,  $A_3$ ,  $C_1$ ,  $C_2$ , and  $C_3$ , while no subgraphs of  $g_2$  were shared. He/she can then compose all obtained subgraphs into a partially connected provenance graph through the overall dependency structure among all uploaded graphs stored in the smart contract PGAC, even when  $g_2$  is missing.

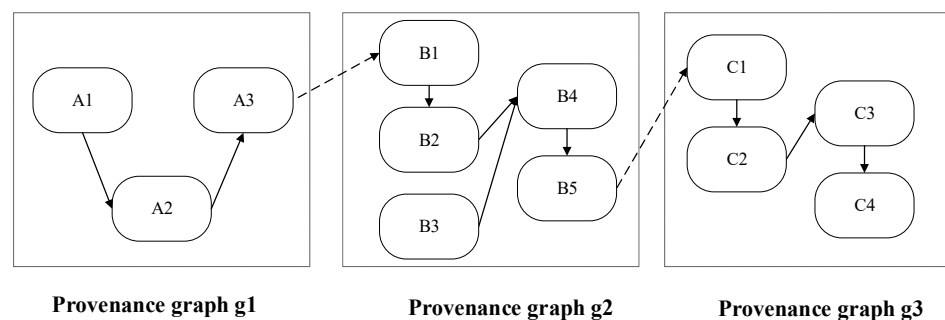


Figure 9. Traceability analysis.

### 8.4. Cost of Storage and Communication

The storage cost is the amount of gas consumed by the blockchain network for storing a provenance graph. Suppose that  $n$  is the number of elements, including all nodes or edges, of a local provenance graph, and that  $m$  is the average number of elements of a subgraph that can be embedded in a transaction. The storage cost of storing each element of the provenance graph on-chain is  $Cost_{element}$ . The cost of each transaction for storing a subgraph is  $Cost_{block}$ . The cost of executing corresponding smart contracts to extract and

establish dependencies among provenance graphs, as well as storing the index information of a local provenance graph, is  $Cost_{smart}$ . The total storage cost  $Cost_{gas}$  can be computed in Formula (1).

$$Cost_{gas} = n * Cost_{element} + \left\lceil \frac{n}{m} \right\rceil * Cost_{block} + Cost_{smart} \tag{1}$$

In practice, when the consortium blockchain starts,  $m$  is usually defined by each peer independently as  $m * c < b$ , where  $c$  is the average size of each element in a provenance graph and  $b$  is the block size limit predefined in the genesis block. The cost of storage  $Cost_{gas}$  varies when the total number of nodes  $n$  and the number of subgraphs  $\lceil \frac{n}{m} \rceil$  changes. In our experiments, we set the parameter  $c$  to 0.4 kb and  $b$  to 40 kb, and the number of elements in a provenance graph  $n$  to 200, 400, or 1000. Then, we explore how the storage cost  $Cost_{gas}$  changes when the number of subgraphs changes. As shown in Figure 10, if the number of nodes in a provenance graph to be uploaded is fixed, the gas consumption for storing it onto the blockchain increases along with an increase in the number of subgraphs in the provenance graph. Thus, less provenance elements embedded in a transaction can lead to a higher overall storage cost. At an extreme, our approach can degenerate to the one that embeds one provenance record in a single transaction. In fact, when the number of subgraphs is low, the main gas consumption is used for storing the provenance graph itself. However, when the number of subgraphs is high, the total gas consumption  $\lceil \frac{n}{m} \rceil * Cost_{block}$  for transactions of uploading a lot of subgraphs becomes non-trivial. In addition, we can see from Figure 10 that when the number of elements increases, the minimal number of subgraphs increases.

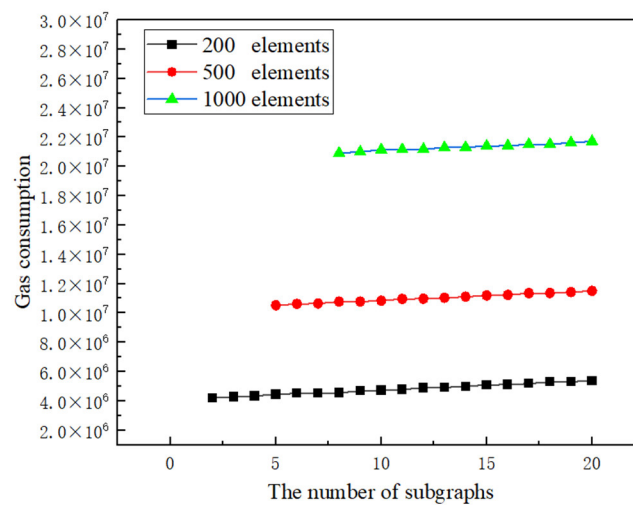


Figure 10. Correlations between the storage cost and the number of subgraphs.

Assume there are  $K$  peers in the provenance sharing network. Each peer needs to partition a provenance graph  $G_i$  with equal  $n$  elements into  $\lceil \frac{n}{m} \rceil$  subgraphs with  $m$  elements. The total number of transactions for uploading these subgraphs is  $K * \lceil \frac{n}{m} \rceil$ . These transactions need to spread across the network in a given time period. Note that if  $K * \lceil \frac{n}{m} \rceil$  is greater than the number of transactions that can be processed by the underlying consortium blockchain in that time period, provenance uploading is delayed. In order to avoid any accumulation of transactions in the transaction pool, one can tune the average block time and transaction gas limit according to domain-specific requirements.

### 8.5. Comparison Analysis

We compare the BSTProv system with two categories of blockchain-based provenance sharing solutions. One is the so-called “single-record-on-chain” solutions that embed each provenance record in a single blockchain transaction; the other is the so-called “hashes-on-chain” solutions that embed the hash value of a provenance graph in a single blockchain transaction while storing the original provenance graph off-chain.



As shown in Table 2, our approach achieves higher trustworthiness and higher security at the relatively higher cost of storage and communication than the “hashes-on-chain” solutions. Note that “hashes-on-chain” solutions can easily conduct flexible authorization, privilege management, and provenance tracing by storing and managing the provenance graph off-chain, for example in some relational databases. However, they suffer from low trustworthiness and security, which cannot be tolerated in trustworthiness-critical scenarios. Our approach achieves trustworthiness and security as high as what the “single-record-on-chain” solutions achieve while significantly saves the cost of storing and spreading a large number of transactions used for enveloping a large number of provenance records. Although the “single-record-on-chain” solutions enables more flexible authorization than our approach, they suffer the complexity of defining and verifying access control policies for a large number of provenance records, and of integrating provenance records scattered in different blockchain transactions to enable global provenance tracing, while our approach addresses these issues by dealing with provenance records in a batch mode.

**Table 2.** A comparison of solutions.

Solution	Trustworthiness	Security	Flexible Authorization	Privilege Management	Traceability	Storage Cost	Communication Cost
Single record on-chain	High	High	High	Hard	Hard	High	$K \times n$
Hashes on-chain	Low	Low	Easy	Easy	Easy	Low	K
Our approach	High	High	Mid	Easy	Mid	Mid	$K \times [n/m]$

## 9. Conclusions and Future Work

This paper presents the BSTProv, i.e., a blockchain-based secure and trustworthy data provenance sharing system. It enables secure provenance sharing by partitioning the local provenance graph into multiple subgraphs and embedding the encrypted subgraphs into blockchain transactions. It enables flexible authorization and cross-domain provenance tracing by storing the overall dependency structures among provenance graphs and the encrypted permissions to different requesters in smart contracts. It enables efficient and trustworthy provenance storage and tracing by a consortium blockchain and local databases. The evaluation experiment showed that the proposed system enables secure and trustworthy provenance sharing with limited storage cost, high dependability, high traceability, and flexible authorization. Future work should optimize a provenance graph partitioning algorithm and enable adaptable and trustworthy re-partitioning to a submitted subgraph.

**Author Contributions:** Conceptualization and methodology, L.-S.S.; software, validation, and formal analysis, C.Z. and Y.L.; writing—original draft preparation and writing—review and editing, L.-S.S. and X.B.; writing—review and editing, Y.-B.Z. and W.-Q.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Gudivada, V.N.; Baeza-Yates, R.; Raghavan, V.V. Big data: Promises and problems. *Computer* **2015**, *48*, 20–23. [[CrossRef](#)]
2. Heck, S.; Bianchini, F.; Souren, N.Y.; Wilhelm, C.; Plass, C. Fake data, paper mills, and their authors: The International Journal of Cancer reacts to this threat to scientific integrity. *Int. J. Cancer* **2021**, *149*, 492–493. [[CrossRef](#)] [[PubMed](#)]
3. Baesens, B.; Höppner, S.; Verdonck, T. Data engineering for fraud detection. *Decis. Support Syst.* **2021**, *150*, 113492. [[CrossRef](#)]
4. Stoldt, J.P.; Weber, J.H. Provenance-based Trust Model for Assessing Data Quality during Clinical Decision Making. In Proceedings of the 2021 IEEE/ACM 3rd International Workshop on Software Engineering for Healthcare (SEH), Madrid, Spain, 3 June 2021; pp. 24–31.
5. Li, Y. The W3C PROV family of specifications for modelling provenance metadata. *Comput. Rev.* **2014**, *55*, 310.

6. Lu, R.; Lin, X.; Liang, X.; Shen, X.S. Secure provenance: The essential of bread and butter of data forensics in cloud computing. In Proceedings of the ACM Symposium on Information, Beijing, China, 13 April 2010; pp. 282–292.
7. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* 2008, p. 21260. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 10 April 2022).
8. Crosby, M.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. *Appl. Innov.* **2016**, *2*, 6–10.
9. Narayanan, U.; Paul, V.; Joseph, S. Decentralized blockchain based authentication for secure data sharing in Cloud-IoT. *J. Ambient Intell. Humaniz. Comput.* **2021**, *13*, 769–787. [[CrossRef](#)]
10. Xu, Z.; Wang, Q.; Wang, Z.; Liu, D.; Wen, S. PPM: A Provenance-Provided Data Sharing Model for Open Banking via Blockchain. In Proceedings of the ACSW '20: Australasian Computer Science Week 2020, Melbourne, Australia, 4–6 February 2020; pp. 1–8.
11. Nakasumi, M. Information sharing for supply chain management based on block chain technology. In Proceedings of the 2017 IEEE 19th Conference on Business Informatics (CBI), Luxembourg, 24–27 July 2017; pp. 140–149.
12. Liang, X.; Shetty, S.S.; Tosh, D.K.; Kamhoua, C.A.; Kwiat, K.A. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Proceedings of the IEEE/ACM CCGRID, Madrid, Spain, 14–17 May 2017; pp. 468–477.
13. Fernando, D.; Kulshrestha, S.; Herath, J.D.; Mahadik, N. SciBlock: A blockchain-based tamper-proof non-repudiable storage for scientific workflow provenance. In Proceedings of the 2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC), Los Angeles, CA, USA, 12–14 December 2019; pp. 81–90.
14. Wood, G. Ethereum: A secure decentralized generalized transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
15. Androulaki, E.; Manevich, Y.; Muralidharan, S.; Murthy, C.; Laventman, G. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the 13th EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
16. Sai, A.R.; Buckley, J.; Fitzgerald, B.; Gear, A.L. Taxonomy of centralization in public blockchain systems: A systematic literature review. *Inf. Process. Manag.* **2021**, *58*, 102584. [[CrossRef](#)]
17. Dib, O.; Brousmiche, K.L.; Durand, A. Consortium blockchains: Overview, applications and challenges. *Int. J. Adv. Telecommun.* **2018**, *11*, 51–64.
18. Zhang, A.; Lin, X. Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *J. Med. Syst.* **2018**, *42*, 1–18. [[CrossRef](#)]
19. Syalim, A.; Nishide, T.; Sakurai, K. Preserving integrity and confidentiality of a directed acyclic graph model of provenance. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Rome, Italy, 21–23 June 2010; pp. 311–318.
20. Porkodi, S.; Kesavaraja, D. Secure Data Provenance in Internet of Things using Hybrid Attribute based Crypt Technique. *Wirel. Pers. Commun.* **2021**, *118*, 2821–2842. [[CrossRef](#)]
21. Missier, P.; Bryans, J.; Gamble, C.; Curcin, V. Abstracting PROV provenance graphs: A validity-preserving approach. *Future Gener. Comput. Syst.* **2020**, *111*, 352–367. [[CrossRef](#)]
22. Danger, R.; Curcin, V.; Missier, P.; Bryans, J. Access control and view generation for provenance graphs. *Future Gener. Comput. Syst.* **2015**, *49*, 8–27. [[CrossRef](#)]
23. Sun, L.; Park, J.; Dang, N.; Sandhu, R. A Provenance-Aware Access Control Framework with Typed Provenance. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 411–423. [[CrossRef](#)]
24. Deutch, D.; Frankenthal, A.; Gilad, A.; Moskovitch, Y. On optimizing the trade-off between privacy and utility in data provenance. In Proceedings of the 2021 International Conference on Management of Data, Xi'an, China, 20–25 June 2021; pp. 379–391.
25. Liu, J.; Li, X.; Ye, L.; Zhang, H.; Du, X.; Guizani, M. BPDFS: A blockchain based privacy-preserving data sharing for electronic medical records. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
26. Wei, P.; Wang, D.; Zhao, Y.; Tyagi, S.K.; Kumar, N. Blockchain data-based cloud data integrity protection mechanism. *Future Gener. Comput. Syst.* **2020**, *102*, 902–910. [[CrossRef](#)]
27. Xia, Q.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access* **2017**, *5*, 14757–14767. [[CrossRef](#)]
28. Rifi, N.; Rachkidi, E.; Agoulmine, N.; Taher, N.C. Towards using blockchain technology for IoT data access protection. In Proceedings of the IEEE International Conference on Ubiquitous Wireless Broadband, Salamanca, Spain, 5 May 2017; pp. 1–5.
29. Peng, K.; Li, M.; Huang, H.; Wang, C.; Choo, K.K.R. Security Challenges and Opportunities for Smart Contracts in Internet of Things: A Survey. *IEEE Internet Things J.* **2021**, *8*, 12004–12020. [[CrossRef](#)]
30. Dai, H.N.; Zheng, Z.; Zhang, Y. Blockchain for Internet of Things: A Survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094. [[CrossRef](#)]
31. Ferrag, M.A.; Lei, S. The Performance Evaluation of Blockchain-based Security and Privacy Systems for the Internet of Things: A Tutorial. *IEEE Internet Things J.* **2021**, *8*, 17236–17260. [[CrossRef](#)]
32. Ramachandran, A.; Kantarcioglu, M. Smart provenance: A distributed, blockchain based data provenance system. In Proceedings of the 8th ACM Conference, Tempe, AZ, USA, 13 March 2018; pp. 35–42.
33. Chen, W.; Liang, X.; Li, J.; Qin, H.; Mu, Y.; Wang, J. Blockchain based provenance sharing of scientific workflows. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 3814–3820.
34. Bose, R.J.; Phokela, K.; Kaulgud, V. Podder Blinker: A blockchain-enabled framework for software provenance. In Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference (APSEC), Putrajaya, Malaysia, 2–5 December 2019; pp. 1–8.

35. Ruan, P.; Dinh, T.T.A.; Lin, Q.; Zhang, M.; Chen, G.; Ooi, B.C. LineageChain: A fine-grained, secure and efficient data provenance system for blockchain. *VLDB J.* **2021**, *30*, 3–24. [[CrossRef](#)]
36. Bischì, A.; Basile, M.; Poli, D.; Vallati, C.; Desideri, U. Enabling low-voltage, peer-to-peer, quasi-real-time electricity markets through consortium blockchains. *Appl. Energy.* **2021**, *288*, 116265. [[CrossRef](#)]
37. Buluc, A.; Meyerhenke, H.; Safro, I.; Sanders, P.; Schulz, C. Recent advances in graph partitioning. *Algorithm Eng.* **2013**, 117–158. [[CrossRef](#)]
38. Zulfiqar, M.; Tariq, F.; Janjua, M.U.; Mian, A.N.; Qayyum, A.; Qadir, J.; Sher, F.; Hassan, M. EthReview: An Ethereum-based Product Review System for Mitigating Rating Frauds. *Comput. Secur.* **2021**, *100*, 102094. [[CrossRef](#)]
39. Nazi, A.; Hang, W.; Goldie, A.; Ravi, S.; Mirhoseini, A. Gap: Generalizable approximate graph partitioning framework. *arXiv* **2019**, arXiv:1903.00614v1.
40. Eberhardt, J.; Tai, S. On or off the blockchain? Insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*; Springer: Cham, Switzerland, 2017; pp. 3–15. [[CrossRef](#)]
41. Hjalmarsson, F.P.; Hreioarsson, G.K.; Hamdaqa, M.; Hjalmtýsson, G. Blockchain-based e-voting system. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 1 July 2018; pp. 983–986.
42. Cheah, Y.W.; Plale, B.; Morwick, J.K.; Leake, D.; Ramakrishnan, L. A noisy 10 GB provenance database. In Proceedings of the Business Process Management Workshops—BPM 2011 International Workshops, Clermont-Ferrand, France, 29 August 2011; pp. 370–381.