Ma *et al. BMC Medical Genomics* 2020, **13**(Suppl 7):91
https://doi.org/10.1186/s12920-020-0725-y

BMC Medical Genomics

RESEARCH Open Access

# Efficient logging and querying for blockchain-based cross-site genomic dataset access audit

Shuaicheng Ma[1*], Yang Cao[2] and Li Xiong[1]

*From* 7th iDASH Privacy and Security Workshop 2018
San Diego, CA, USA. 15 October 2018

## Abstract

**Background:** Genomic data have been collected by different institutions and companies and need to be shared for broader use. In a cross-site genomic data sharing system, a secure and transparent access control audit module plays an essential role in ensuring the accountability. A centralized access log audit system is vulnerable to the single point of attack and also lack transparency since the log could be tampered by a malicious system administrator or internal adversaries. Several studies have proposed blockchain-based access audit to solve this problem but without considering the efficiency of the audit queries. The 2018 iDASH competition first track provides us with an opportunity to design efficient logging and querying system for cross-site genomic dataset access audit. We designed a blockchain-based log system which can provide a light-weight and widely compatible module for existing blockchain platforms. The submitted solution won the third place of the competition. In this paper, we report the technical details in our system.

**Methods:** We present two methods: baseline method and enhanced method. We started with the baseline method and then adjusted our implementation based on the competition evaluation criteria and characteristics of the log system. To overcome obstacles of indexing on the immutable Blockchain system, we designed a hierarchical timestamp structure which supports efficient range queries on the timestamp field.

**Results:** We implemented our methods in Python3, tested the scalability, and compared the performance using the test data supplied by competition organizer. We successfully boosted the log retrieval speed for complex AND queries that contain multiple predicates. For the range query, we boosted the speed for at least one order of magnitude. The storage usage is reduced by 25%.

**Conclusion:** We demonstrate that Blockchain can be used to build a time and space efficient log and query genomic dataset audit trail. Therefore, it provides a promising solution for sharing genomic data with accountability requirement across multiple sites.

**Keywords:** Blockchain, Genome, Cross-site genomic datasets, Access log audit

*Correspondence: sma30@emory.edu
[1]Department of Computer Science, Emory University, 400 Dowman Dr, Atlanta, GA, USA
Full list of author information is available at the end of the article

© The Author(s). **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

## Background

With the rapid development of biomedical and computational technologies, a large amount of genomic data sets have been collected and analyzed in national and international projects such as Human Genome Project [1] , the HapMap project [2] and the Genotype-Tissue Expression (GTEx) project [3], which yielded invaluable research data and extended the boundary of human knowledge. Thanks to the advance of computer technology, the cost of genomic testing is dropping exponentially. Nowadays, the testing price ranges from under $100 to more than $2,000, depending on the nature and complexity of the test [4]. One can test her gene easily and cheaply by using services from DNA-testing companies such as Ancestry and 23andMe. Given the above, genomic data sets have been scattered around the world in different institutions and companies. On the other hand, the potential business value of genomic data and privacy concerns [5–7] hinder the sharing of cross-sites genomic data. Notably, the General Data Protection Regulation (GDPR) restricts the exchange of personal data. Under GDPR, such sensitive data only could be accessed after obtaining the consent of data subjects (i.e., the one who owns the data) and providing accountability audit. This requires that any cross-site genomic data sharing system should be equipped with a secure and transparent access control module.

Blockchain technology has received increasing attention because it provides a new paradigm of value exchange. Although it stems from cryptocurrency, many studies have investigated the adoption of blockchain in different application scenarios beyond financial domain that typically involve multiple parties with conflict of interests such as personal data sharing [8–10], supply chain [11–13], identity management [14, 15] and medical data management [16–22]. They show that using blockchain technology can reduce friction and increase transparency. A blockchain system has several notable features: decentralization, immutability and transparency. These are achieved by cryptographic hash, consensus algorithm and many other innovations from previously unrelated fields such as cryptography and distributed computation [23]. Due to the space limitation, we do not introduce more details of blockchain technologies and refer interested readers to surveys on blockchain [24–31].

Several studies investigated blockchain-based access log audit [32–34] (we introduce them in the next section). They focus on how to achieve the immutability of the log. However, none of them investigated the efficiency of logging and querying for a blockchain system at the application layer. On the other hand, a few recent studies [35–38] from database community consider a blockchain system as a distributed database, and attempt to improve the performance of such system by exploring new designs of bottom layers (such as storage or transaction processing) of the system. However, without considering the application characteristics, such modifications on the back-end engine of the system may not have the desired performance improvement on every application or even cause unexpected side effects.

The 2018 iDASH competition first track, "*Blockchain-based immutable logging and querying for cross-site genomic dataset access audit trail*", provides us with an opportunity to explore a light-weight and widely compatible access audit module for existing blockchain platforms. Our submitted solution won the third place of the competition. In this paper, we report the system design and technical details in our solution.

### The competition task [39]

The goal of iDASH competition 2018 first track is to develop blockchain-based ledgering solutions to log and query the user activities of accessing genomic datasets across multiple sites. Concretely, given a genomic data access log file in which each entry includes seven attributes including Timestamp, Node, ID, Ref − ID, User, Activity, Resource, the task is to design a time/space efficient data structure and mechanisms to store and retrieve the logs based on Multichain version 1.0.4 [40].

**Competition setup and requirement.** It is required that each entry in the data access log must be saved individually as one transaction (i.e., participants cannot save the entire file in just one transaction), and all log data and intermediate data (such as index or cache) must be saved on-chain (no off-chain data storage allowed). Competition participants can determine how to represent and store each log entry in transactions. It does not need to be a plain text copy of the log entry. Also, the query implementation should allow a user to search the log using any field of one log entry (i.e., node, id, user, resource, activity, timestamp, and a "reference id" referring to the id of the original resource request), any "AND" combination (e.g., node AND id AND user AND resource), and any timestamp range (e.g., from 1522000002418 to 1522000011441) using a command-line interface. Also, the user should be able to sort the returning results in ascending/descending order with any field (e.g., timestamp). There will be 4 nodes in the blockchain network, and 4 log files to be stored. Users should be able to query the data from any of the 4 sites. Participants can implement any algorithms to store, retrieve, and present the log data correctly and efficiently.

**Evaluation Criteria.** The logging/querying system needs to demonstrate good performance (i.e., accurate query results) by using a testing dataset, which is different from the one provided for the participants. The speed, storage/memory cost, and scalability of each solution will

be evaluated. The competition organizer used the binary version of Multichain 1.0.4 on 64-bit Ubuntu 14.04 with the default parameters as the test bed for fairness. No modification of the underlying Multichain source code is allowed. The submitted executable binaries should be non-interactive (i.e., depend only on parameters with no input required while it works), and should contain a readme file to specify the parameters. The organizer tested all submissions using 4 virtual machines, each with 2-Core CPU, 8GB RAMs and 100GB storage.

### Related work

The closest line of work to this competition is blockchain-based access log audit. Suzuki et al., [32] proposed a method using blockchain as an audit-able communication channel. This study is motivated by a similar problem studied in this paper: in a client-server system, the logging on either server-side or client-side does not provide strict means of auditing, because the host of the logging system could tamper the log. They implemented a proof-of-concept system on top of Bitcoin by encoding the messages (i.e., API calls from clients and Replies from the server) between clients and the server into the transactions of bitcoin. Since the transactions are publicly available, they can be retrieved and verified by an auditor as needed. The proposed system is easy to use and convenient for a client-server system. However, answering the audit query using the proposed system may be time-consuming, especially for a large-scale system serving millions of clients, as each reply is returned in the form of a bitcoin transaction. The maximum transaction processing capacity of bitcoin is estimated between 3.3 and 7 transactions per second [41].

Castaldo et al., [33] implemented a blockchain-based tamper-proof audit mechanism for OpenNCP (Open National Contact Points) [42], which is a system for exchanging eHealth data between countries in Europe. The idea is similar to the one proposed in [32], but dealing with data exchange instead of answering queries. They also encode the data that need to be exchanged into the transactions, but the data are encrypted using symmetric keys which are shared in advance between the sender and receiver through a secure channel. The author suggests to use Multichain because it provides low overhead for the transactions handling.

ProvChain [34] is a blockchain based data provenance architecture for assuring data operation (i.e., data access and data changes) in the cloud storage application. This differs from the previous two solutions mentioned, as the major challenge is that the provenance data are also sensitive but still need to be validated by a third party. The authors proposed an additional layer as provenance auditor which interacts with a blockchain network by blockchain receipts which include provenance entry for future validation.

The 2018 iDASH competition first track provides us with an opportunity to explore the design of efficient logging and querying methods for a blockchain system. We attempt to design a blockchain-based log system that can serve as a light-weight and widely compatible component for the existing blockchain platforms. Especially, our solution is optimized for genomic dataset access auditing under the requirements of the competition task.

## Method

We design a blockchain-based log system that is time/space efficient to store and retrieve genomic dataset access audit trail. Our method only leverages the Blockchain mechanism and is not limited to any specific Blockchain implementation, such as Bitcoin[43], Ethereum[44]. We introduce an on-chain indexing data structure which can be easily adapted to any blockchains that use a key-value database as their local storage. In our development, we use Multichain version 1.0.4 as an interface between Bitcoin Blockchain and our insertion and query method. Multichain is a Bitcoin Blockchain fork. It conveniently provides a feature, data stream, to allow us to use Bitcoin Blockchain as an append-only key-value database.

### Overview

In Fig. 1, we illustrate the overview of the logging system, which is built on top of Multichain APIs. The core task is to design space and time efficient methods for insertion and queries. As described in Section "Technical details of the task", there are three types of primitive queries: point query, AND query, and range query. There are seven fields in the given genomic dataset: Timestamp, Node, ID, Ref − ID, User, Activity, Resource as shown in Table 1.

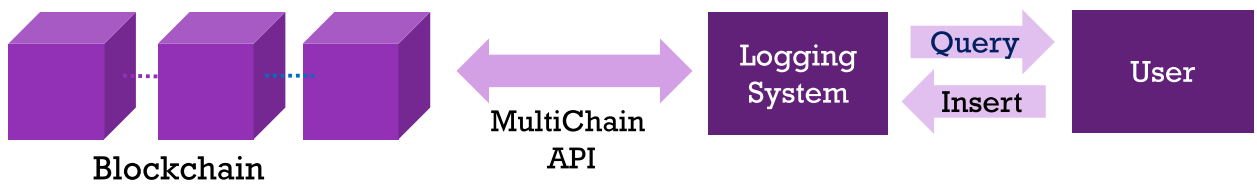


**Fig. 1** Overview of the logging system

**Table 1** The Sample Logs

| Timestamp | Node | ID | Ref-ID | User | Activity | Resource |
|---|---|---|---|---|---|---|
| 152202801 | 1 | 1 | 1 | 1 | REQ_RESOURCE | MOD_FlyBase |
| 152208352 | 1 | 2 | 1 | 1 | VIEW_RESOURCE | MOD_FlyBase |
| 152216966 | 1 | 3 | 3 | 6 | FILE_ACCESS | GTEx |
| 152237149 | 1 | 9 | 9 | 10 | REQ_RESOURCE | MOD_SGD |

For point query, the user can query on any field. For AND query, the user can query on any combination of fields. For range query, the user can query only on timestamp field with a start and end timestamp. See Table 1 & 2 as a running example.

### Baseline method

We first describe a naive method as a baseline. The baseline method leverages only three Multichain APIs as shown in Table 3.

**Insertion:** First, we create $K$ streams, where $K$ is the number of fields. Multichain builds $K$ tables in its backend key-value database. Second, we build $K$ key-value pairs, where the key is the attribute data and the value is entire record line. Finally, we convert those $K$ pairs into one Blockchain transaction and publish it to Blockchain. The following Fig. 2 is an example conversion from a log record to blockchain transaction. We will use this example log record in the remaining sections. After the transaction is confirmed by Blockchain, Multichain decodes the transaction and insert each key-value pairs to its corresponding table.

**Point Query:** The implementation of a point query is straightforward which simply returns a list of records as shown in Algorithm 1. In this literature, we assume the run time complexity of all Multichain API is $O(1)$. The run time complexity of Point Query is $O(1)$.

**Algorithm 1:** Point Query

**Input**: $A$, $K$ //attribute and key
**Output**: $l_r$ //a list of record
1 $l_r \leftarrow liststreamkeyitems(A, K)$
2 **return** $l_r$

**Table 2** Insertion and Queries Examples

Insertion

- *Insert*("152202801 1 1 1 1 REQ_RESOURCE MOD_FlyBase")

Queries

- *Point_Query*(Activity="VIEW_RESOURCE")
- *AND_Query*(ID="2", Node="1")
- *Range_Query*(start=1522000000000, end=1522000100000)

**Table 3** Multichain APIs Used in Our Methods

| Multichain APIs | Description |
|---|---|
| create[ stream name] | create a stream(table) in database |
| publish[ streamname] [ key] [ value] | Insert key-value pair to specific stream(table) |
| liststreamkeyitems[ stream name] [ key] | Retrieve all items with the given key |

**AND Query:** AND query enables a user to query with multiple keys. We convert AND query to multiple point queries and intersect the result of all point queries. The run time complexity is $O(K)$, where $K = number\ of\ keys$.

**Algorithm 2:** AND Query

**Input**: $l_{AK}$ // a list of attribute and key pairs
**Output**: $l_r$ //a list of record
1 $l_r \leftarrow point\_query(l_{AK}[0]_A, l_{AK}[0]_K)$
2 **foreach** $(A, K) \in l_{AK}$ **do**
3 $\quad l_r \leftarrow l_r \cap point\_query(A,K)$
4 **return** $l_r$

**Timestamp Range Query:** Given a start timestamp and an end timestamp, Timestamp Range Query returns records whose timestamp is in this range. We convert Timestamp Range Query into R point queries, where R is the range of timestamp. The run time complexity is $O(R)$, where $R = range\ of\ timestamp$.

**Algorithm 3:** Timestamp Range Query

**Input**: $t_s$, $t_e$ // start timestamp and end timestamp
**Output**: $l_r$ //a list of record
1 $l_r \leftarrow \{\}$
2 **for** $t = t_s$ ***to*** $t_e$ **do**
3 $\quad l_r \leftarrow l_r \cup point\_query$("Timestamp",t)
4 **return** $l_r$

### Enhanced method

After testing the baseline solution, which will be discussed in the result section, we found that the retrieve speed heavily depends on the number of API calls. Therefore, the fewer API calls we use, the faster retrieve speed we get. More specifically, we found three non-optimal issues:

- The entire record is duplicated K times where K is the number of fields, which is insufficient in terms of storage overheads.
- Since we need to query all results and intersect them in local memory, AND query takes significant

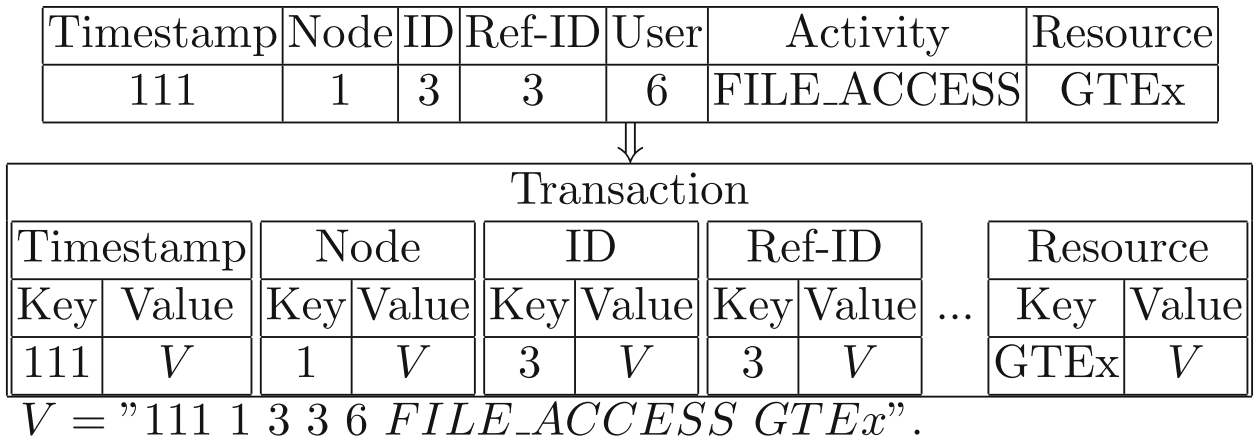


**Fig. 2** Log Record to Transaction Conversion

amount of memory when the number of AND operations increases.
- If the length of a given range query is $n$ (typically, $n$ is ranging from $10^6$ to $10^8$), the baseline method naively translate the range query into $n$ point queries and concatenate the results.

The blockchain-based auditing system is an append-only structure, so a data structure that keeps the minimum amount of information while maintaining the efficiency is essential. The percentage of read(query) operations in the real-world auditing system is low [45], therefore we trade retrieval speed for storage cost. We redesign the key-value pairs in the blockchain transaction, modified the query algorithm accordingly and built a selectivity list based on data distribution. Most of all, we design a hierarchical timestamp structure which significantly reduces the number of queries(APIs) needed for the range query.

**Insertion:** To address these problems, we redesign the key-value pairs. The key part remained the same (attribute data), but we removed the entire entry from the value part. As a result, we removed all duplicated values in the baseline method as shown in Fig. 3

**Point Query:** Since we now have an empty value in the key-value database, we cannot use the key to get original record directly. We now take advantage of Blockchain transaction ID which is included in the returning JSON file of *liststreamkeyitems* API. First, we get a list of *TXID* (transaction ID) with the given key. Second, we use another Multichain API, *getrawtransaction*, to get the matching transactions. Finally, we rebuild the original record from the transaction where all attribute data are included. It is worth mentioning that the point query now requires 1 + T times API calls to retrieve the records where T is the size of the *TXID* list. If the modification is allowed in this competition, we can combine these three steps into one, which reduces the total API calls from 1 + T to 1. In other words, if Multichain nodes can perform the work from line 3 to line 6 in Algorithm 4, users can point query with just one API call. The run time complexity of our point query is $O(T)$, where T is the size of the *TXIDs* list.

**AND Query:** In order to reduce the retrieval cost, we build a selectivity list for attributes based on the example test data which was given by the competition organizer. A selectivity list is based on the rank of result size of each attribute. The attribute which has the smallest query

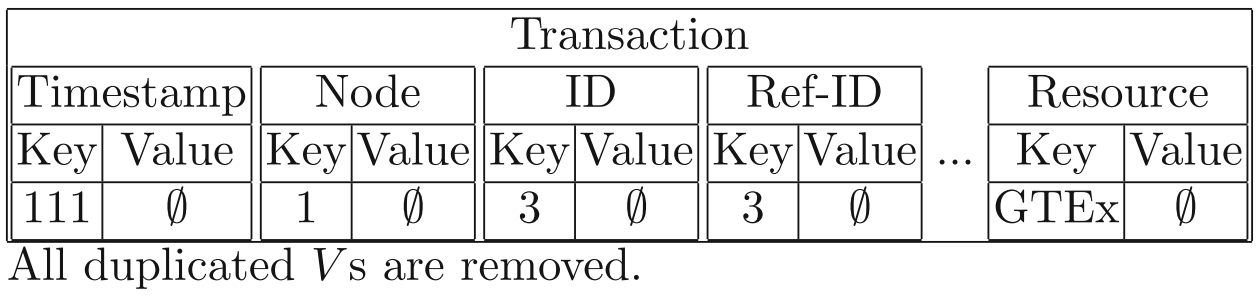


**Fig. 3** Transaction with empty values

**Algorithm 4:** Point Query with additional step

**Input**: $A, K$ //attribute and key
**Output**: $l_r$ //a list of record
1 $[\ TXIDs] \leftarrow liststreamkeyitems(A, K)$
2 $l_r \leftarrow [\ ]$
3 **foreach** $txid \in [\ TXIDs]$ **do**
4 $\quad T \leftarrow getrawtransaction\ (txid)$
5 $\quad R \leftarrow rebuild(T)$
6 $\quad append(l_r, R)$
7 **return** $l_r$

result size is the most selective. For the enhanced AND query, we call point query only one time for the most selective key then filter the result in the memory. Since we only query once from Blockchain, the total memory usage is bounded by the largest query result. The run time complexity is $O(1)$.

**Algorithm 5:** AND Query with selectivity list

**Input**: $l_{AK}, l_S$ // a list of attribute and key pairs and a selectivity list
**Output**: $l_r$ //a list of record
1 $SK \leftarrow findMostSelectiveKey(l_{AK}, l_s)$
2 $l_r \leftarrow point\_query(SK_A, SK_K)$
3 **foreach** $(A, K) \in l_{AK}$ **do**
4 $\quad l_r \leftarrow filter(l_r, A, K)$

**Timestamp Range Query:** Since Blockchain is an immutable structure, the common indexing techniques, such as B-tree and R-tree, which require adjusting/balancing the entire data structure according to the data distribution, won't work. We introduce a hierarchical timestamp structure, which is an incremental data structure and matches the append-only characteristics of the blockchain system. Our design significantly reduces the number of queries(APIs) needed for a single range query.

The hierarchical timestamp structure consists of multiple levels. See Table 4 as an example. The range in the high level divides into multiple smaller range in the lower level. We denote each range part as LevelNumber:Starting Timestamp. A timestamp is recorded in the corresponding part at all levels. In our running example, a timestamp *111* will be recorded in L0:100, L1:110, and L2:111 in Table 4.

**Table 4** Simple Hierarchical Timestamp Structure

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| L0 | [100,200) | | | | | | | | |
| L1 | [100,110) | | | [110,120) | | | [120,..) | | |
| L2 | 100 | ... | 109 | 110 | ... | 119 | 120 | ... | ... |

To build this structure, we need to slightly modify the insertion method by adding L streams where L is the number of levels, and we need to add L key-value pairs to Blockchain transaction as well. See Fig. 4 as an example.

In our enhanced range query method, we recursively find the largest range in the hierarchical timestamp structure and use multiple point queries to retrieve the result.

**Algorithm 6:** Timestamp Range Query with hierarchical timestamp structure

**Input**: $t_s, t_e$ // start timestamp and end timestamp
**Output**: $l_r$ //a list of record
1 $l_r \leftarrow list$
2 $l, r \leftarrow findLargestRange(t_s, t_e)$
3 **while** $r \neq None$ **do**
4 $\quad append(l_r, point\_query(l, r))$
5 $\quad l, r \leftarrow findLargestRange(t_s, t_e)$
6 **return** $l_r$

In the following example, we show the number of queries(APIs) needed for our baseline range query and enhanced range query.

Range query from timestamp 109 to timestamp 120.

Baseline Method:

$q('T', 109) \cup q('T', 110) \cup ... \cup q('T', 120) \rightarrow 11$ queries

Enhanced Method:

$q('L2', 109) \cup q('L1', 110) \cup q('L2', 120) \rightarrow 3$ queries

We reduce the number of queries needed for range query from $R_{T_e - T_s}$ to $\sum_{i=0}^{L} \frac{R_i}{r_{L_i}}$, where $R_{i+1} = R_i \bmod r_{L_i}$, $R_0 = R_{T_e - T_s}$ and $r_{L_i}$ is the elemental range at level $L_i$.

The run time complexity of the enhanced range query is $O\left(\sum_{i=0}^{L} \frac{R_i}{r_{L_i}}\right)$.

#### *Further optimizations*

The database normalization can be used for both baseline and enhanced solution. According to the given datasets, *Ref-ID* refers back to the same original *ID* which means those *User* and *Resource* are the same. For this reason, we can exclude *User* and *Resource* in Blockchain transaction.

## Results

### Implementation environment

We used Python3 as our main programming language to develop our solution, Savior [46] to interact with Multichain API and Docker [47] to simulate 4 Blockchain nodes. Additionally, we created some bash scripts to automatically setup Blockchain nodes and Multichain environment. We also wrote a benchmark program to compare our baseline method and enhanced method. Our code

| Transaction | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Timestamp | | ... | Resource | | L0 | | L1 | | L3 | | |
| Key | Value | | Key | Value | Key | Value | Key | Value | Key | Value | |
| 111 | ∅ | | GTEx | ∅ | 100 | ∅ | 110 | ∅ | 111 | ∅ | |

**Fig. 4** Transaction with Hierarchical Timestamp Structure

is available online [48]. The specifications of our testing machine are as follows: 6 cores CPU(i7 8700k), 32 GB of RAM and 6TB of HDD with Ubuntu 16.04 as the operating system.

We used the sample testing data supplied by the competition organizer to benchmark our implementation. The sample testing data consists of 4 files, one per node. Each file has $10^5$ entries of log records which has 7 fields (Timestamp, Node, ID, Ref − ID, User, Activity, Resource). To illustrate, we provide a few sample data in Table 1.

To find the optimal number of levels and the step multiplier of two adjacent levels for the hierarchical timestamp structure (Fig. 4), we test all reasonable parameter combinations by brute-force. For the given sample data, the optimal parameter for the number of levels is 3 and the step multiplier of two adjacent levels is 100. Future work may include finding the optimal number of levels in a more efficient way.

## Benchmark

In our benchmark experiment, we show the scalability of our two methods alongside LevelDB[49] as a reference. Many blockchain systems[43, 44, 50] use LevelDB as a back-end database to store the raw transaction data. It is worth mentioning that those systems only index the raw transactions, not the actual content inside the transactions. Database system and Blockchain do not share the same design goal: the former is usually administered by a centralized entity, and the latter intents to work in a trustless environment. Nevertheless, this comparison offers useful insights of Blockchain based log system which trades speed for data integrity. We simulate the enhanced insertion, the enhanced point query, and the enhanced AND query behavior in LevelDB. For range query, we use LevelDB native method so we can properly examine our hierarchical timestamp structure. In all tests, we run 10 rounds for each methods with respect to varying the number of records. We calculate the average and the standard deviation from the results. We notice that the standard deviation is extremely small which shows the little trace in all figures expect Fig. 5(Point Query). This is due to the identical environment and the setup of our simulated blockchain nodes.

### *Scalability test: queries*

Figure 5 shows query time with respect to the varying number of records for point query, range query, AND query. For the point query test, the response time is determined by the result size. As the number of records increases, the result size increases and the response time increases. The response time of the enhanced method is worse than the baseline method because of the addition API calls which we introduced in the enhanced point query. For the range query test, the performance

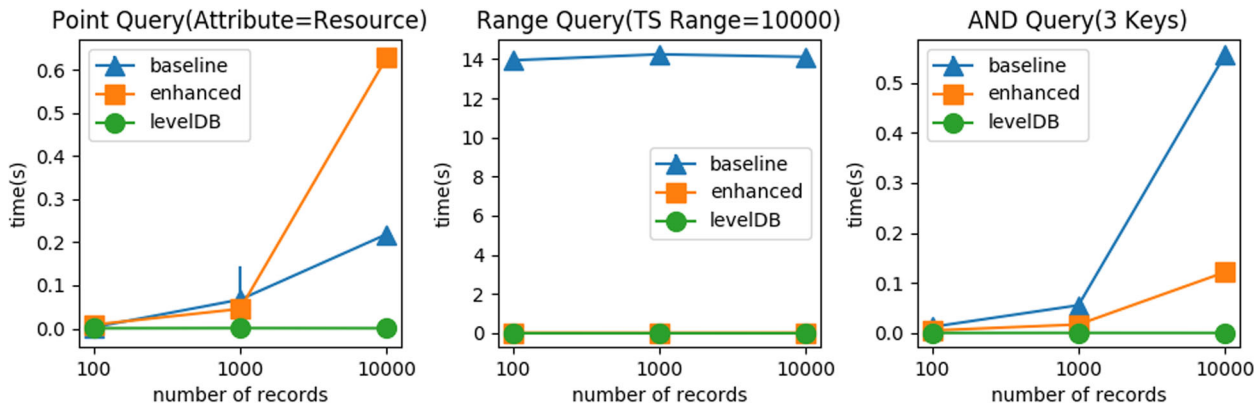


**Fig. 5** Scalability Test: Queries

is constant since the result size of certain time range is constant. It is worth mentioning that our enhanced range query method have very close performance comparing to the native LevelDB range query method. For AND query, since it consists of point query, the response time increases with the increasing number of records. It is worth mentioning that the selectivity list design in our enhanced AND query method offsets the drawback of the enhanced point query method when the number of keys is larger than 2.

#### *Scalability test: insertion*

Figure 6 shows the completion time of insertion methods with respect to varying the number of records. The insertion time is depended on the transaction size. The insertion times of the two methods are approximately the same. The enhanced method needs more key-value pairs to support hierarchical timestamp indexing structure. However, the empty values in key-value pairs offset this transaction size increment.

#### *Scalability test: storage*

Figure 7 shows the total blockchain size in bytes with respect to varying the number of records. The blockchain size information is collected by calling Multichain API. Since Blockchain and LevelDB measure their size in different ways, we exclude LevelDB in this test. The figure suggests that the enhanced method uses less storage than the baseline method. The duplication removal from the blockchain transaction in the enhanced method works as designed.

#### *Detailed comparison*

In this section, we show a detailed performance difference of 3 query types in the baseline method, enhanced method, and LevelDB. We use the fixed 1000 records in the remaining tests.

**Point Query:** Figure 8 shows the query response time for different attributes. The enhanced method performance is worse than the baseline method, because of the additional API calls in the enhanced method. The rank in the result also matches the rank in selectivity list which indicates the return record size. The return record size of *Activity* is the largest among the attributes. In other words, *Activity* has the lowest selective and need more API calls to get the result than other attributes, so it has the worst performance difference.

**Range Query:** Figure 9 shows the query response time with respect to varying the time range. The enhanced method is at least one order of magnitude better than the baseline method. It proves that our hierarchical timestamp structure can batch a large number of queries into a small (almost constant) number of queries. Hence, the enhanced method achieves almost constant time performance as LevelDB native range query method.

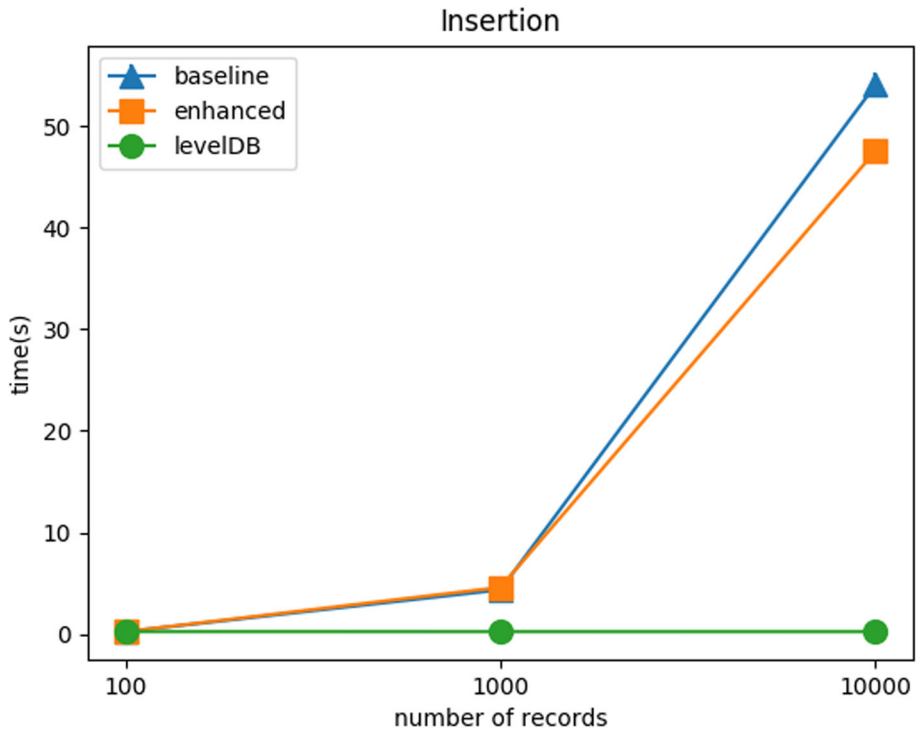


**Fig. 6** Scalability Test: Insertion

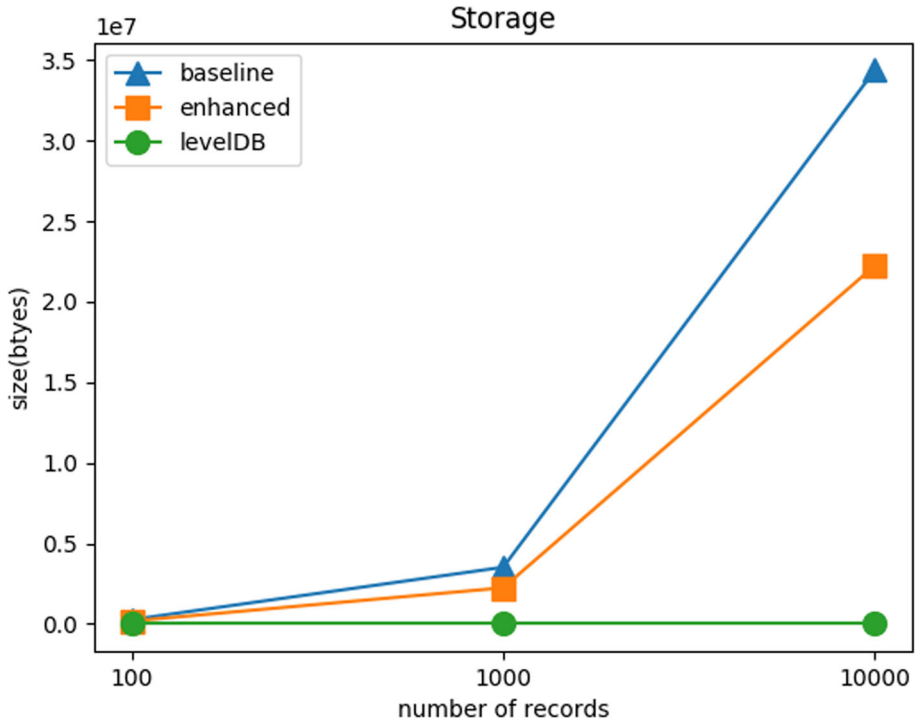


**Fig. 7** Scalability Test: Storage

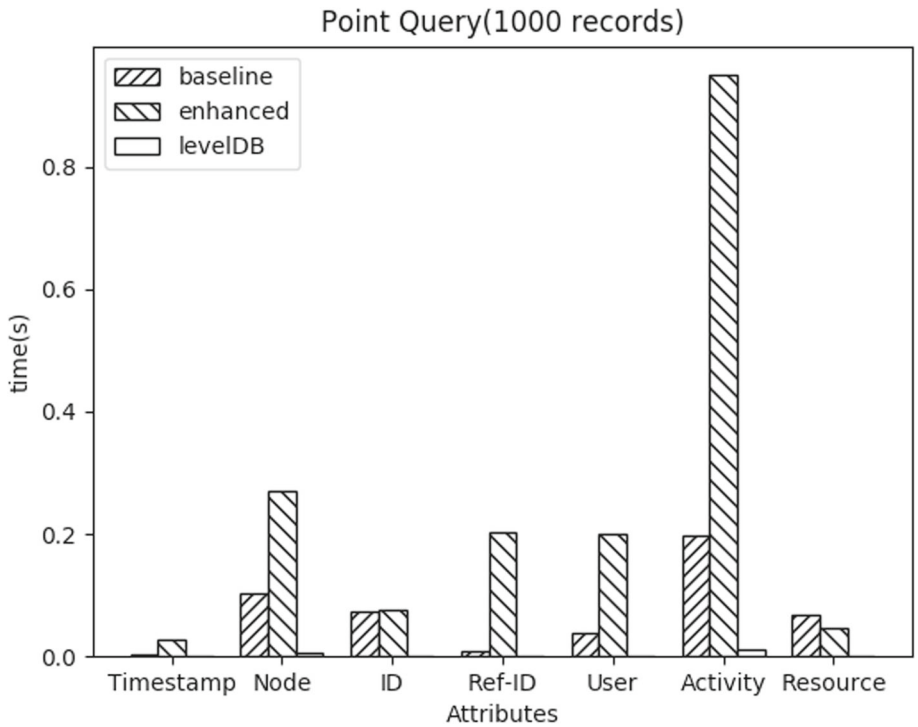


**Fig. 8** Point Query

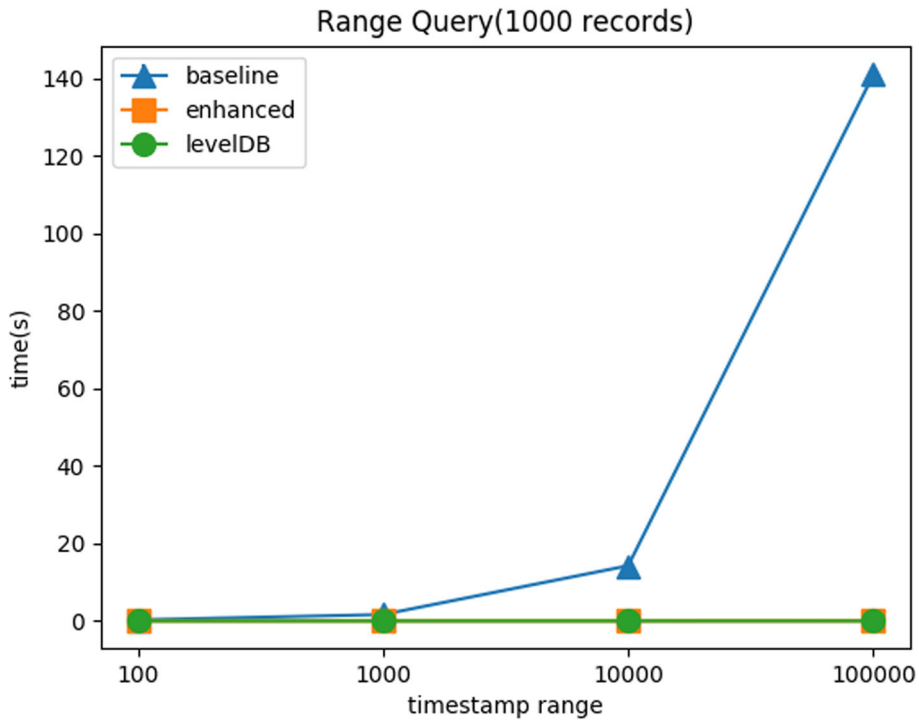


**Fig. 9** Range Query

**AND Query:** Figure 10 shows the query time with respect to varying the number of keys. We test all combinations of keys. For example, for 2 keys test, we test all 21 combinations(7 choose 2) and average the result. It is much easier to find a more selective key when the number of keys is increasing. This is the reason why the enhanced method has a downward slope. When there are only 2 keys, the enhanced method has high possibility to find a low selective key. As a result, when AND query takes a low selective key, it requires a long response time.

## Discussion

Our design is heavily governed by the competition requirements, evaluation criteria[39], and Multichain 1.0.4 capability. In this paper, we intend to use Multichain as only an interface, so our design can be applied to any arbitrary blockchain system. Multichain 1.0.4 does not allow an item to have multiple keys and competition does not allow participators to modify Multichain, so we have to manually construct the blockchain transaction. There are two major developments for the future work: 1) A new interface which encodes/decodes the log entry to/from blockchain transaction more efficient. For example, in Bitcoin blockchain transaction script, it can write entire log entry only once in the blockchain transaction and let local interface client translate the script to the database. A specific Bitcoin interface for this log system can significantly reduce the transaction size. 2) A new Blockchain oriented database system, such as Forkbase[37]. It aims to design a new key-value architecture to reduce the development efforts of the above application and provide efficient analytical query performance. It is possible to replace the key-value engines in the existing blockchain platforms for better query performance.

In this paper, we focus on designing efficient logging and querying schemes for immutable blockchain systems, and assume the blockchain network has been well-established under a specific *consensus algorithm* and acceptable *transaction throughput*. In the following, we discuss how they may affect our solution.

Consensus algorithm may affect the performance of insertion functions because a newly generated access log (as a transaction) need to be accepted by all node in the network (achieving a consensus on the next block) in order to be stored in the ledger. Consensus algorithm manifests the transaction throughput, which is majorly controlled by a predefined parameter in Multichain called mining-diversity (the default configuration is 0.3). If the transaction throughput is low, the insertion would be insufficient since it may be suspended until the previous batch of logs is finished. The transaction throughput also affects the audit queries because the query is performed on the locally synchronized ledger. Under low transaction throughput, a newly generated log may take a long time to

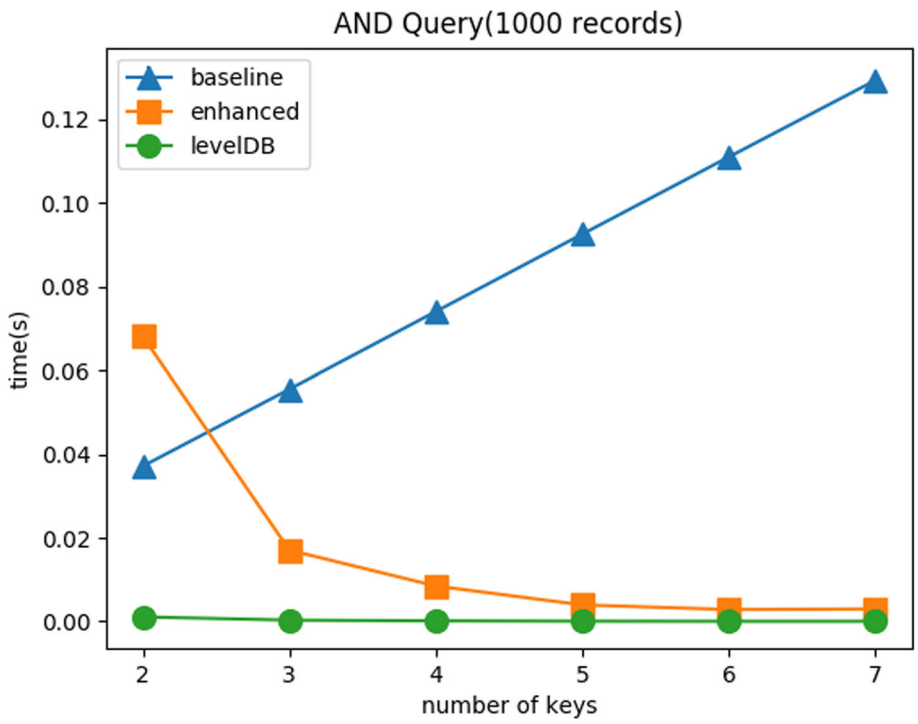


**Fig. 10** AND Query

be included in the ledger and synchronized to a node so that the query on a node may not be able to provide the accurate real-time answer.

Further, the access log could be private since it records all of the queries issued by a user. This is a challenge for existing blockchain platforms since the ledger is public to every node in the network for increasing transparency and security. A recent version of Hyperledger Fabric [50] includes a new function for this problem. The idea is dividing the ledger to different channels and selectively sharing a channel among a subset of users. There are also other efforts for this problem by adopting secure multiparty computation [9], zero-knowledge proof [10] or trusted hardware [51]. Although this problem is beyond the scope of this competition, our solution could be extended using the above techniques.

## Conclusions

In this paper, we presented two solutions for blockchain-based logging and querying genomic dataset audit trail. We built a baseline solution and then adjusted our implementation based on the evaluation criteria of the competition[39] and the general real-world characteristics of log systems[52]. The blockchain-based log system is an append-only structure, so the storage increases monotonically. In the real world, the percentage of writing operation(insertion) is much higher than the portion of reading operation(query) in the workload [52]. Based on the above two reasons, we decided to prioritize the storage space over retrieval speed and insertion speed. We can reduce the storage cost by 25% and increase the range query speed by at least one order of magnitude. We claim that our hierarchical timestamp structure design is Blockchain implementation independent. It can be adapted to any Blockchain (e.g., Bitcoin, Ethereum, Hyperledger) with the help of an intermediary, such as Multichain.

### Abbreviations

GDPR: General data protection regulation; API: Application program interface; JSON: JavaScript object notation; TXID: Transaction identification; GTEx: Genotype-tissue expression

### Acknowledgements

We thank Dr. Tsung-Ting Kuo, the organizer of iDASH competition 2018 first track, for providing informative Q&A and helpful advice!

### Author's contributions

SM, YC and LX designed the solution and wrote the manuscript. SM implemented the code. All authors read and approved the final manuscript.

### About this supplement

This article has been published as part of BMC Medical Genomics Volume 13 Supplement 7, 2020: Proceedings of the 7th iDASH Privacy and Security Workshop 2018. The full contents of the supplement are available online at https://bmcmedgenomics.biomedcentral.com/articles/supplements/volume-13-supplement-7.

### Funding

This work was supported by NIH R01GM118609, Georgia CTSA under grant NIH UL1TR002378, NSF under grant CNS-1618932, the AFOSR DDDAS program under grant FA9550-121-0240, the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research No. 17H06099, No. 18H04093 and

No. 19K20269. The publication costs were funded by NIH R01GM118609, Georgia CTSA under grant NIH UL1TR002378, Japan Society for the Promotion of Science (JSPS) No. 17H06099, No. 18H04093, No. 19K20269.

**Availability of data and materials**
The data that support the findings of this study are available from the iDash workshop. The code is available at Github(https://github.com/mshuaic/Blockchain_med/).

**Ethics approval and consent to participate**
Not applicable.

**Consent for publication**
Not applicable.

**Competing interests**
The authors declare that they have no competing interests.

**Author details**
[1]Department of Computer Science, Emory University, 400 Dowman Dr, Atlanta, GA, USA. [2]Department of Social Informatics, Kyoto University, Kyoto, Japan.

Published: 21 July 2020

**References**

1. Collins FS, Morgan M, Patrinos A. The human genome project: lessons from large-scale biology. Science. 2003;300(5617):286–290.
2. Consortium IH. The international HapMap project. Nature. 2003;426(6968):789.
3. Lonsdale J, Thomas J, Salvatore M, Phillips R, Lo E, Shad S, Hasz R, Walters G, Garcia F, Young N, Foster B, Moser M, Karasik E, Gillard B, Ramsey K, Sullivan S, Bridge J, Magazine H, Syron J, Fleming J, Siminoff L, Traino H, Mosavel M, Barker L, Jewell S, Rohrer D, Maxim D, Filkins D, Harbach P, Cortadillo E, Berghuis B, Turner L, Hudson E, Feenstra K, Sobin L, Robb J, Branton P, Korzeniewski G, Shive C, Tabor D, Qi L, Groch K, Nampally S, Buia S, Zimmerman A, Smith A, Burges R, Robinson K, Valentino K, Bradbury D, Cosentino M, Diaz-Mayoral N, Kennedy M, Engel T, Williams P, Erickson K, Ardlie K, Winckler W, Getz G, DeLuca D, MacArthur D, Kellis M, Thomson A, Young T, Gelfand E, Donovan M, Meng Y, Grant G, Mash D, Marcus Y, Basile M, Liu J, Zhu J, Tu Z, Cox NJ, Nicolae DL, Gamazon ER, Im HK, Konkashbaev A, Pritchard J, Stevens M, Flutre T, Wen X, Dermitzakis ET, Lappalainen T, Guigo R, Monlong J, Sammeth M, Koller D, Battle A, Mostafavi S, McCarthy M, Rivas M, Maller J, Rusyn I, Nobel A, Wright F, Shabalin A, Feolo M, Sharopova N, Sturcke A, Paschal J, Anderson JM, Wilder EL, Derr LK, Green ED, Struewing JP, Temple G, Volpi S, Boyer JT, Thomson EJ, Guyer MS, Ng C, Abdallah A, Colantuoni D, Insel TR, Koester SE, Little AR, Bender PK, Lehner T, Yao Y, Compton CC, Vaught JB, Sawyer S, Lockhart NC, Demchok J, Moore HF. The genotype-tissue expression (GTEx) project. Nat Genet. 2013;45:580–5.
4. Wetterstrand KA. DNA sequencing costs: data from the NHGRI genome sequencing program (GSP). 2013. www.genome.gov/sequencingcostsdata. Accessed 1 June 2020.
5. Malin BA, Emam KE, O'Keefe CM. Biomedical data privacy: problems, perspectives, and recent advances. J Am Med Inf Assoc. 2013;20(1):2–6.
6. Gkoulalas-Divanis A, Loukides G, Sun J. Publishing data from electronic health records while preserving privacy: A survey of algorithms. J Biomed Inf. 2014;50:4–19.
7. Naveed M, Ayday E, Clayton EW, Fellay J, Gunter CA, Hubaux J-P, Malin BA, Wang X. Privacy in the genomic era. ACM Comput Surv. 2015;48(1):6–1644.
8. Zyskind G, Nathan O, Pentland A. Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops. IEEE; 2015. https://doi.org/10.1109/spw.2015.27.
9. Zyskind G, Nathan O, Pentland A. Enigma: Decentralized computation platform with guaranteed privacy. arXiv:1506.03471 [cs]. 2015.
10. Froelicher D, Egger P, Sousa JS, Raisaro JL, Huang Z, Mouchet C, Ford B, Hubaux J-P. UnLynx: a decentralized system for privacy-conscious data sharing. Proc Priv Enhancing Technol. 2017;2017(4):232–50.
11. Hackius N, Petersen M. Blockchain in logistics and supply chain : trick or treat?. In: Kersten WB, Thorsten R, Christian M, editors. Digitalization in Supply Chain Management and Logistics: Smart and Digital Solutions for an Industry 4.0 Environment. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 23. Berlin: epubli GmbH; 2017. p. 3–18. http://hdl.handle.net/10419/209299. https://doi.org/10.15480/882.1444.
12. García-Bañuelos L, Ponomarev A, Dumas M, Weber I. Optimized execution of business processes on blockchain. In: Lecture Notes in Computer Science; 2017. p. 130–46. https://doi.org/10.1007/978-3-319-65000-5_8.
13. Abeyratne SA, Monfared RP. Blockchain ready manufacturing supply chain using distributed ledger. Int J Res Eng Tech. 2016;5(9):1–10.
14. Azouvi S, Al-Bassam M, Meiklejohn S. Who am i? secure identity registration on distributed ledgers. In: Lecture Notes in Computer Science; 2017. p. 373–89. https://doi.org/10.1007/978-3-319-67816-0_21.
15. Yasin A, Liu L. An online identity and smart contract management system. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). IEEE; 2016. https://doi.org/10.1109/compsac.2016.2.
16. Kuo T-T, Ohno-Machado L. ModelChain: decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. arXiv:1802.01746 [cs]. 2018.
17. Yue X, Wang H, Jin D, Li M, Jiang W. Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. J Med Syst. 2016;40(10):218.
18. Xia Q, Sifah EB, Asamoah KO, Gao J, Du X, Guizani M. MeDShare: trust-less medical data sharing among cloud service providers via blockchain. IEEE Access. 2017;5:14757–67.
19. Azaria A, Ekblaw A, Vieira T, Lippman A. MedRec: using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD). IEEE; 2016. https://doi.org/10.1109/obd.2016.11.
20. Genestier P, Zouarhi S, Limeux P, Excoffier D, Prola A, Sandon S, Temerson J-M. Blockchain for consent management in the ehealth environment: A nugget for privacy and security challenges. J Int Soc Telemed eHealth. 2017;5:24.
21. Choudhury O, Sarker H, Rudolph N, Foreman M, Fay N, Dhuliawala M, Sylla I, Fairoza N, Das AK. Enforcing human subject regulations using blockchain and smart contracts. Blockchain Healthc Today. 2018. https://doi.org/10.30953/bhty.v1.10.
22. Li C, Cao Y, Hu Z, Yoshikawa M. Blockchain-based bidirectional updates on fine-grained medical data. In: 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW). IEEE; 2019. https://doi.org/10.1109/icdew.2019.00-40.
23. Narayanan A, Clark J. Bitcoin's academic pedigree. Commun ACM. 2017;60(12):36–45.
24. Kuo T-T, Kim H-E, Ohno-Machado L. Blockchain distributed ledger technologies for biomedical and health care applications. J Am Med Inf Assoc. 2017;24(6):1211–20.
25. Underwood S. Blockchain beyond bitcoin. Commun ACM. 2016;59(11):15–7.
26. Sun J, Yan J, Zhang KZK. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. Financ Innov. 2016;2(1):26.
27. Wörner D, von Bomhard T, Schreier Y-P, Bilgeri D. The bitcoin ecosystem: Disruption beyond financial services? 2016.
28. Bonneau J, Miller A, Clark J, Narayanan A, Kroll JA, Felten EW. SoK: research perspectives and challenges for bitcoin and cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy. IEEE; 2015. https://doi.org/10.1109/sp.2015.14.
29. Tschorsch F, Scheuermann B. Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Commun Surv Tutorials. 2016;18(3):2084–123.
30. Pilkington M. Blockchain technology: principles and applications. Research handbook on digital transformations. 2016225–253. https://doi.org/10.4337/9781784717766.00019.
31. Zheng Z, Xie S, Dai H, Chen X, Wang H. An overview of blockchain technology: Architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress). IEEE; 2017. https://doi.org/10.1109/bigdatacongress.2017.85.
32. Suzuki S, Murai J. Blockchain as an audit-able communication channel. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). IEEE; 2017. https://doi.org/10.1109/compsac.2017.72.

33. Castaldo L, Cinque V. Blockchain-based logging for the cross-border exchange of eHealth data in europe. In: Gelenbe E, Campegiani P, Czachórski T, Katsikas SK, Komnios I, Romano L, Tzovaras D, editors. Security in Computer and Information Sciences. Cham: Springer International Publishing; 2018. p. 46–56. 978-3-319-95189-8.
34. Liang X, Shetty S, Tosh D, Kamhoua C, Kwiat K, Njilla L. ProvChain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE; 2017. https://doi.org/10.1109/ccgrid.2017.8.
35. Dinh TTA, Liu R, Zhang M, Chen G, Ooi BC, Wang J. Untangling blockchain: A data processing view of blockchain systems. IEEE Trans Knowl Data Eng. 2018;30(7):1366–1385.
36. Dinh TTA, Wang J, Chen G, Liu R, Ooi BC, Tan K-L. BLOCKBENCH: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data. New York: Association for Computing Machinery; 2017. p. 1085–100. 9781450341974, https://doi.org/10.1145/3035918.3064033.
37. Wang S, Dinh TTA, Lin Q, Xie Z, Zhang M, Cai Q, Chen G, Ooi BC, Ruan P. Forkbase: an efficient storage engine for blockchain and forkable applications. Proc VLDB Endowment. 2018;11(10):1137–50.
38. Xu Z, Han S, Chen L. CUB, a consensus unit-based storage scheme for blockchain system. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE; 2018. https://doi.org/10.1109/icde.2018.00025.
39. iDASH Secure Genome Analysis Competition 2018. GMC Med Genomics. 2019. http://www.humangenomeprivacy.org/2018/. Accessed 1 June 2020.
40. MultiChain Private Blockchain White Paper. https://www.multichain.com/download/MultiChain-White-Paper.pdf. Accessed 4 June 2019.
41. Croman K, Decker C, Eyal I, Gencer AE, Juels A, Kosba A, Miller A, Saxena P, Shi E, Sirer EG, Song D, Wattenhofer R. On scaling decentralized blockchains. In: Clark J, Meiklejohn S, Ryan PeterYA, Wallach D, Brenner M, Rohloff K, editors. Financial Cryptography and Data Security. Berlin: Springer Berlin Heidelberg; 2016. p. 106–25. 978-3-662-53357-4.
42. Fonseca M, Karkaletsis K, Cruz IA, Berler A, Oliveira IC. OpenNCP: a novel framework to foster cross-border e-health services. Stud Health Technol Inf. 2015;210:617–21.
43. Bitcoin. https://bitcoin.org/en/. Accessed 4 June 2019.
44. Ethereum. https://www.ethereum.org/. Accessed 4 June 2019.
45. Roselli D. Characteristics of file system workloads. USA: University of California at Berkeley; 1998.
46. A Python Wrapper for Multichain Json-RPC API. https://github.com/DXMarkets/Savoir. Accessed 4 June 2019.
47. Docker. https://www.docker.com/. Accessed 4 June 2019.
48. Our Code at Github. https://github.com/mshuaic/Blockchain_med. Accessed 4 June 2019.
49. LevelDB. https://github.com/google/leveldb. Accessed 4 June 2019.
50. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S, Murthy C, Nguyen B, Sethi M, Singh G, Smith K, Sorniotti A, Stathakopoulou C, Vukolić M, Cocco SW, Yellick J. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. New York: Association for Computing Machinery; 2018. p. 30. 9781450355841, https://doi.org/10.1145/3190508.3190538.
51. Hynes N, Dao D, Yan D, Cheng R, Song D. A demonstration of sterling: A privacy-preserving data marketplace. Proc VLDB Endow. 2018;11(12):2086–9.
52. Rosenblum M, Ousterhout JK. The design and implementation of a log-structured file system. ACM Trans Comput Syst. 1992;10(1):26–52. https://doi.org/10.1145/146941.146943.


## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Ready to submit your research? Choose BMC and benefit from:**

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

**At BMC, research is always in progress.**

**Learn more** biomedcentral.com/submissions