A Decentralized System for Medical Data Management via Blockchain

Qingzhu Yang[1,2], Qiao Liu[1,2], Hairong Lv[1,2,3]
[1] Department of Automation, Tsinghua University, China
[2] Bioinformatics Division, Beijing National Research Centre for Information Science and Technology, China
[3] AIHealthX, Inc., China
yangqz@tsinghua.edu.cn, liu-q16@mails.tsinghua.edu.cn, lvhairong@tsinghua.edu.cn

Abstract

With the rapid advance of data science, massive medical data are managed in data centers which may cause surveillance and security issues. This calls for building a secure data management system. In this paper, we propose a decentralized system for medical data management, to address the challenge of data privacy protection during data sharing between network nodes, leveraging the blockchain technology. We propose protocols to achieve this and normalize data format recorded on the blockchain. The entities of our system include users, contributing data; edge data hub (EDH) nodes, used for storing users' data (an EDH stores one user's data and is controlled by the corresponding user); service nodes, with the ability to analyze user data and customers that need the analysis of the data. To protect data privacy in the EDH, we use a secure computing platform (SCP) to process data. Furthermore, data request from a service with high credibility is more likely to be permitted by the EDH. Data authenticity is guaranteed by checking the hash value recorded on the blockchain. Moreover, we reduce the off-blockchain storage space leveraging the data sparsity. Our system thus enables data sharing, authenticity, privacy protection and storage space reduction, shedding lights on building a developable and secure data-driven world.

Keywords: Blockchain, Medical data, Data sharing, Data privacy, Storage

1 Introduction

Massive data are continually being collected and dissected, resulting in innovation and growth of data economics [1]. Data are collected and analyzed by organizations and companies for personalizing services, predicting the future direction and more [1]. Data are important substances in the economy [2]. There is also an unprecedented development in human medical data, including phenomic, anatomic, physiologic and biologic information [3]. Human can now be quantified by methods of medical examination and treatments, scans, sensors, sequencing and laboratory tests [4]. Medical data are critical for both treatment and research. It is reported that public health and biomedical researchers have been analyzing data to capacitate precision medicine to exploit new treatments and control public health risks from a lot of sources [5].

With the development of the data-promoted society, user privacy has become a growing public concern [1]. At present, the medical data are mostly stored, managed and distributed by a few large centralized data centers. Unfortunately, the centralized data storage based on data centers has a lot of privacy and security issues. As the data centers take full control of all the data they stored, the data owner lose most or even all the control over the data [1]. Moreover, if a hacker gets access to the data center, he can access all the data simultaneously and the whole system may break down [6].

In order to conquer various issues of the centralized data storage systems, several methods have been developed. The decentralized systems are explored where data are stored and managed by independent nodes. Data are retrieved by authorized queries and transferred through the internet based on encryption. However, there is not a consistent view of current status in these systems which enforces the authorized key to be forever stored [6]. Recently, blockchain has been investigated, allowing the participants to exchange information securely through a publicly verifiable ledger, without a data center [7]. This system has been successfully implemented as a cryptocurrency known as the bitcoin [1, 8]. It has been proposed to apply blockchain technologies to overcome the forementioned barriers. There are already several blockchain-based systems that were proposed for managing medical data. Medrec [9] is a decentralized record management system to handle electrical medical records (EMRs) based on the blockchain technology. It can manage authentication, confidentiality and data sharing when handling sensitive information. MeDShare [10] is another blockchain-based system that addresses the issue of medical data sharing among

*Corresponding Author: Hairong Lv; E-mail: lvhairong@tsinghua.edu.cn
DOI: 10.3966/160792642020092105010

in a trust-less environment. The design of MeDShare employs the smart contracts and an access control mechanism. In [11], the authors proposed a secure and trustable EMR data management and sharing system using the blockchain technology. This is one of the few works to implement a prototype that has been proven to ensure privacy and fine-grained access control. Although several blockchain-based systems were proposed for the purpose of medical data sharing or management, many previous works lack a detailed description of the technical protocol. In the majority of current studies, some form of data will be delivered from the request node after the request is permitted. The solution of the data leakage needs further investigation. Meanwhile, the efficient use of storage needs to be considered when the capacity is inadequate.

In this paper, leveraging blockchain technology, we propose a decentralized data management system focusing on medical data sharing, privacy protection, and storage saving, which contains both the blockchain and off-blockchain storage. In this system, medical data are stored in edge data hubs (EDHs), and each EDH is fully controlled by its owner who contributes the data. The public blockchain is adopted in our system. The Proof of Work (PoW) algorithm is used for consensus [8, 12-13]. Data access is permitted by the owner of each EDH. The organizations, such as gene companies, could access the data via the permission by the owner of each EDH. The EDH can return the data or the computed result run by the algorithm provided by the service through the secure computing platform (SCP). Thus data privacy and algorithm privacy can be protected by the SCP. The hash values of the data are also stored on the blockchain against tampering. Detailed protocols for data management and normalized formats of data stored on the blockchain are demonstrated in our system. To reduce the storage space, an off-blockchain storage method based on the sparse theory is used. Our proposed system can effectively prevent data leakage when the data analysis request is permitted and reduce the off-blockchain storage while maintaining as much medical data information as possible. Our system demonstrates a framework including data sharing, access permission, data protection, data tamper-proofing and data storage space reduction via the blockchain technology and the sparse theory. We present our work not only as a significant solution for medical data management, but also an important attempt to build a medical data network for secure and open data sharing in the society leveraging the blockchain technology.

2 Related Work

2.1 Centralized Data Management

Centralized data management in authentication and authorization has been investigated recently [14-15]. These schemes depend on a completely-trusted central server to perform authentication, and authentication services. Considering the rapid development of medical systems and the extreme increase of medical data, it is not feasible to deal with large amounts of data by a centralized server. High latency or blocking may be caused by overloaded queries [6, 16].

Moreover, the accident of a single point is a big issue for a centralized network. The more data, the more likely it is to be the target of a hacker attack. If the system is occupied by an attacker, the authorization of the whole system will be destroyed.

2.2 Decentralized Data Management

To deal with the problems of centralization, decentralized systems for establishing trust relationships, authentication and authorization are investigated. In the Web of trust [17] and CCN [18], a decentralized peer-to-peer trust model is performed. In this model, a node, which is identified by a public key, can deliver a signature of another public key for demonstrating authority for trust. The identity does not rely on any central authority. However, a uniform scope of the current status cannot be built in these systems. For instance, a Web of Trust key server cannot testify the existence of a revocation as a key is always reserved [6]. [19] proposed secure multiparty computation-based protocols for collaborative computing of medical data while protecting each participant's privacy. It does not rely on the blockchain technology, lacking the advantage of blockchain-based consensus and non-tampering. [20] demonstrated a data management framework with fine-grained access control. It also described a way to give the calculated answers to the service questions instead of the metadata. It lacked the advantage of blockchain. The SCP in our work is similar in concept to the SafeAnswers in it. In our work, the SCP is closely tied to the blockchain, service protocols, and transactions in our system. We also provide related executing protocols and a complexity analysis for the SCP execution.

2.3 Decentralized Data Management via Blockchain

Recently, blockchain has played an increasingly important role in decentralized data management [9]. In [1], blockchain protocols for data authorization management are used, ensuring fine-grained access control. In [4], the blockchain based technologies for helping manage the use of medical data is suggested. However, no summary of technical work or detailed method is demonstrated in this work. In [6, 9-10, 21], blockchain is used for managing data. However, although access control for data is considered, some form of data will be delivered if data request is

permitted and no further protection is designed to prevent data leakage. [22] implemented the blockchain for data management with access control. A sandbox model is used to isolate the applications running in the personal data center from the personal data. The applications are mainly used as personal tools such as communication software, rather than controlling the data flowing from the personal data center. [23] also proposed a system for data management leveraging the blockchain with fine-grained access control. [9] proposed a blockchain based approach for medical data permission management. They record entities, entity relations, and data permissions, while an off-blockchain store are used to store the raw data. In our work, we also use the blockchain to record entities, and store the real data in an off-blockchain manner. But our system also considers data analysis, privacy protection in the data analysis stage, and recording of the whole interactions between entities, credibility mechanisms. These are integrated into an inseparable system.

In this paper, we refer to the previous ideas and exploit our innovative work towards data sharing, data storage space reduction, data privacy protection and data tamper-proofing via blockchain and sparsity-based compression.

3 System Overview

For our system, the major entities that constitute our system are (1) users, who own data and can update it to the EDHs; (2) edge data hub (EDH) nodes for storing data from the corresponding users; (3) service nodes, the entities (usually organizations) with the ability to analyse personal data, such as the hospitals, gene companies and more, (4) customers, the entities that need the analysis of personal data, as shown in Figure 1. We adopt the public blockchain in our system. The block content includes the data hash values, the transactions, the credibility scores, and so on (section 4.1.2 gives the details).

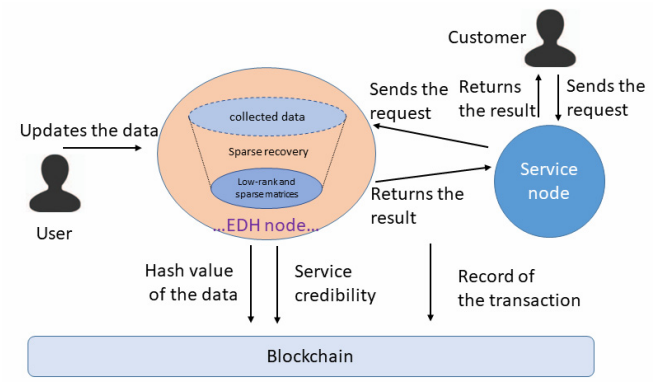

Figure 1. The scheme of the decentralized system for data management

Our framework is designed as follows. The user uploads the data to the corresponding EDH. The EDH maintains a key-value store for the collected data. The collected user data can be compressed to reduce the storage space in the EDH. The hash values of the related data in the EDH are also recorded on the blockchain, hence guaranteeing the data integrity. A customer can request a service for some analysis. The service then retrieves personal data from the EDH, either directly or by SCP execution, depending on the EDH's policy, then performs the analysis, and then returns it to the customer. A typical diagram for the node interaction is shown in Figure 2. The whole transaction process will be recorded on the blockchain. After the process, all the participants can give credibility scores to each other on the blockchain, e.g. based on the quality of analysis. Section 4.4 gives the details. Compared with some previous blockchain based systems where the data permissions are recorded on the blockchain, in our system the permissions are dynamically determined by the EDH, which is more flexible.

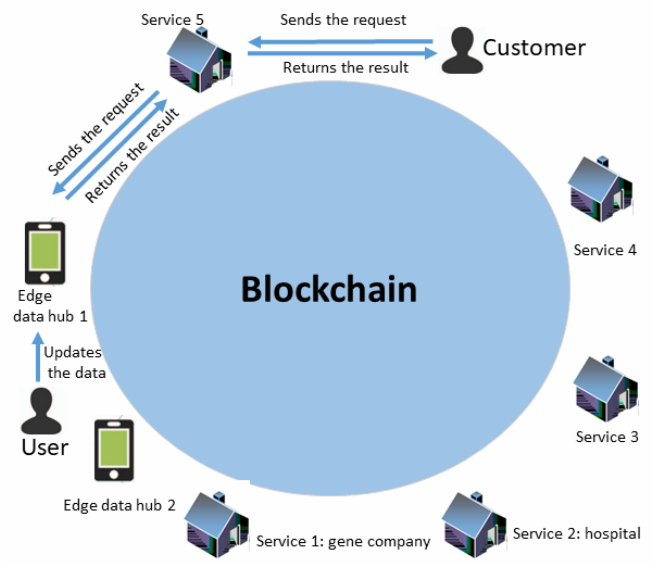

Figure 2. A schematic diagram that shows the network nodes share data using blockchain to protect privacy

In the following sections, we describe the designed protocols of our system.

4 The Network Protocol

4.1 Building Blocks

4.1.1 Entities

The related entities include users, EDH nodes, service nodes and the customers.

The user represents the individual in the network. Each user can store his data on his own EDH node, and can also enjoy services from the service nodes. Each user holds a private/public key pair (U_{sk}, U_{vk}) and is uniquely identified by U_{vk}.

The EDH node is a reliable storage server for the

user. Each EDH node corresponds to a user and is fully controlled by the user. The data stored in EDH can be viewed as a collection of *key-value* pairs, and can be indexed using keys. The key is a string similar to the unix path, e.g. /EHR/hospital/20101010, and the value is a tuple of the form (DataTypeGUID, Data, MetadataTypeGUID, Metadata). The value description is given in section 4.1.2, where the Data/Metadata items are replaced by DataHash/MetadataHash. The EDH nodes are usually deployed in reliable servers, and one reliable server can run several EDH instances. In fact, each EDH node is uniquely related to the user by his public key U_{vk}.

Service nodes provide various services in the network, e.g. data analysis, data providing. Each service node holds a private/public key pair (S_{sk}, S_{vk}) and is uniquely identified by S_{vk}. A service is usually run by a known organization, e.g. a company.

The customer represents the individual or the organization, requesting a service to provide the health data analysis. Each customer holds a private/public key pair (C_{sk}, C_{vk}) and is uniquely identified by C_{vk}.

4.1.2 Blockchain Content

The blockchain is used to record and formulate the elements related to the network, consisting of a train of blocks. Blocks are appended to the blockchain according to a consensus among most nodes of the system. As shown in Figure 3, each block contains a block header and body [16, 23-24]. Each block header is actually a standard blockchain header [16, 23-24].

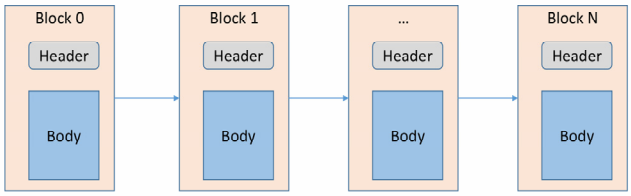

Figure 3. Structure of the blockchain

In this paper, to normalize the data format recoded on the blockchain, meaningful and signed records with predefined schemas are stored in the body of the block. The records in our paper are analogous to the transactions in the body mentioned in [16, 23-24], to avoid ambiguity with the transaction records described in section 4.1.2. The EDH node reads the records to determine if a service is permitted to a special key or not. The users or services can create their own data type declaration to support new types of data. To ensure data consistency, the users or services can store the hash values of private data through a type of record. The general form of a record is as follows:

The **record header** includes the record id, public key of the record writer, the writing time and the record type. The record id identifies each record and is generated automatically by the blockchain.

The **record body** is the content of the record, and the actual organization of the content depends on the record type.

The record is always written by a user, an EDH, a service or a customer, signed with the corresponding private key (with a special signature field). There are several record types.

The **identity declaration** record includes the name and a human readable description of the identity, and the public key of the identity is already in the record header. The identity may be the user, the service or the customer. In our system, each user, service and customer has a unique identity and every identity informs about its existence by the way of declaration. In this way, we do not actually distinguish the user, the service or the customer at the identity level. An identity may be either a user, a service, or a customer depending on its activities. Note that the identity declaration is always the first record of a role (a user, a service, or a customer), which is guaranteed by the blockchain.

An official organization can declare itself as an entity, such as the FDA (U.S. Food and Drug Administration). And the public keys of these organizations should be well known.

The **EDH declaration** record includes the public key of the EDH for a user whose public key is already revealed in the header, and the address (e.g. URL) of the EDH. A user can declare his EDH address multiple times, and the latest declaration overrides the previous ones.

The **data type declaration** record includes the globally unique identifier of the data type (GUID), and a human readable description. Note that the public key of the type publisher is already shown in the header. The data type publisher is usually an official organization (such as the FDA). Standard medical items such as the height, the weight, or various medical tests, should have their own types and be published by standard organization entities (such the FDA). The data type declaration can actually represent any type of data (including both the meaning and data schema) as well as how to use them. The data type GUIDs should not repeat, which is guaranteed by the blockchain.

The **data hash** record is an array of key-value pairs corresponding to the key-value store in the EDH, as described in section 4.1.1. The value has the tuple form (DataTypeGUID, DataHash, MetadataTypeGUID, MetadataHash), where DataTypeGUID refers to the GUID declared in the data type declaration record which specifies what the data means, DataHash is the hash value of the actual data, and MetadataHash corresponds to the metadata containing some additional data, e.g. when and where the tests were taken, whose type (can be viewed as the schema of the data) is also declared in a data type declaration record, and is identified by MetadataTypeGUID (which is also declared in the data type declaration record). To ensure

against tampering, if the service is permitted to the data, it will calculate the data hash value and compare with the hash value recorded on the blockchain. The data hash value record must be written by the corresponding user or its latest EDH. Note that a user may write records for the same key multiple times, and only the last one is considered valid.

The **transaction** record includes the parent transaction record id (PTRI), the transaction type, the transaction data type GUID and the transaction data hash. The transaction data type GUID refers to the GUID declared in the data type declaration record. Each interaction between the components of the system is recorded as one or more transactions on the blockchain. A complete transaction process may consist of several steps (e.g. a custom request a service for analysis), and each step is recorded with a transaction record. These steps are linked with PTRI, where the first step have a null PTRI. Transaction type depends on the application. For example, when a customer requests the service, it writes a REQ_SERVICE transaction with a null PTRI and proper transaction data hash conforming to the type GUID. When the service completes the request, it writes a COMPLETE_SERVICE transaction with PTRI pointing to the previous one, and the data hash of the returned query result. And the customer writes an ACK_COMPLETE_SERVICE to acknowledge its reception of the analysis result. During the data acquisition process with an EDH, the service and the EDH may stagger the REQ_DATA/REQ_SCP, ACK_DATA/ACK_SCP, and RET_DATA/RET_SCP transaction records (see section 4.4 for details).

The **credibility score** record includes public key of the identity it gives, the credibility score, and an array of transaction ids the score based on. Participants in a transaction can score each other, as discussed in section 4.4.

4.1.3 Secure Computing Platform

The service may request data directly from the EDH or request to execute a data analysis program (the SCP program) on the EDH，depending on the EDH local policy. The former is more flexible, but has the risk of leaking the user's privacy. In the latter case the process is executed on the SCP (Secure Computing Platform). The SCP is designed as follows. The service needs to prepare the executable code to analyze data and provide it to run in the EDH. The file can only access authorized personal information. The verified processed results can also be returned back to the service by the restricted execution environment. The actual implementation may refer to mature systems such as the JavaScript in the browser. During execution, when the code wants to access EDH data in the form of a key, e.g. weight of the people, it needs to call a special built-in function. The execution environment then executes the real data access through the data access agent. The data access agent will check the privilege, and reject or return the real data. Section 4.4 gives more details.

4.2 Storage Method

We utilize a method to reduce the off-blockchain storage cost leveraging the sparsity of the data.

Suppose that a user collects a type of health data (e.g. heart health indicators) over time, and the result for each time can be described by a vector D, then we have a series of vectors $D_1, D_2, \ldots, D_T$. We first encode each D_i to get a sparse and low-dimensional vector $E_i, i = 1, 2, \ldots, T$ using the sparse autoencoder method [25], and the sequence $E_i, i = 1, 2, \ldots, T$ to get a matrix M whose i^{th} column is E_i, and then we can compress M leveraging the sparse theory [26]. For self-containment we briefly describe the two major compressing steps, and the reader may refer to [25] and [26] for details.

The first step associates an *encoder-decoder* pair with each $D_i (i = 1, 2, \ldots)$. Each *encoder-decoder* pair corresponds to a type of health data, and we assume a particular health data type. The encoded user data E_i can be obtained as $Encode(D_i)$, which ensures that the dimension of $Encode(D_i)$ is much smaller than the dimension of D_i, that is, $\dim(Encode(D_i) \ll \dim(D_i))$. The decoder is used for recovering encoded data while maintaining the original information as much as possible $(Decode(Decode(D_i) \approx D_i)$. The sparse autoencoder takes the advantages of sparse coding which makes most of nodes in hidden layers under inactive states by adding constraints to the outputs of hidden layers. Assume that the loss function of traditional autoencoder is $JT(W, b)$, where W, b are the model parameters (following the notations in [25]). The loss function of the sparse autoencoder is defined as $J_{sparse}(W, b) = J(W, b) + \beta \sum_{i=1}^{n} KL(\rho \| \hat{\rho}_i)$ [25]. Note that β is the sparsity penalty weight, n is the number of hidden units, term $\hat{\rho}_i$ refers to the average activation of the autoencoder's hidden unit i, ρ is a constant which controls the sparsity, and is usually set to a small value (e.g. $\rho = 0.05$ as in [25]), $KL(\rho \| \hat{\rho}_i)$ is the Kullback-Leibler (KL) divergence between two Bernoulli random variables with mean ρ and $\hat{\rho}_i$ respectively [25]. The KL divergence is a common function to measure how similar two distributions are, and in our case it reveals the sparseness of the activations. We adopt the outputs of hidden layers as the sparse encoded representation $E_i = Encode(D_i)$ regarding the input user data D_i. E_i is usually more

sparse than D_i as it contains lots of zeros, and can be regarded as an informative and efficient transformation of the original data. The *encoder-decoder* pairs are computed and stored in the EDH.

We further transform the encoded sequence $E_i, i = 1, 2, \ldots, T$ in a more efficient way by the low rank and sparse matrix recovery. We concatenate them to a matrix M as described above, and our goal is to represent $M = A + K$, where K is the error matrix and assumed to be even more sparsely supported than M, and A is a low rank matrix. Storing these matrices will consume less space than the original matrix M. The solving process of low rank matrix A and error matrix K can be regarded as an optimization problem through the RPCA method, and the optimization problem can be summarized as $\min_{A,K} rank(A) + \gamma \| K \|_0, subj\ M = A + K$ where γ is the sparsity control coefficient [26]. The reader may refer to [26] for the optimization details.

Once the low rank matrix A is obtained, it can be easily represented by multiplication of two low-dimension matrices which can save storage space. The sparsity of matrix K can also help save storage space. The second step is optional as sometimes we only have data for one or only a few time points.

Figure 4 shows an example of the storage content on and off the blockchain. When a user updates some type of health data for many time points to the EDH, it computes the compressed format and only store the low rank matrix A and the sparse error matrix K.

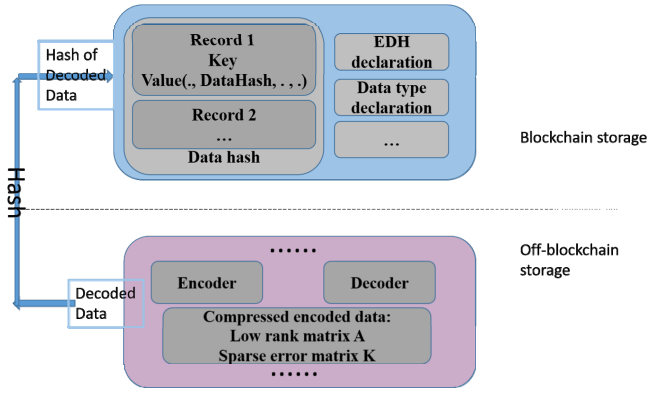

Figure 4. Date storage

The compression is transparent to the clients, and on the blockchain only the hash of the decoded data is stored. The user or the service does not need to know the exitance of compression.

We only apply the sparsity based compressed storage to the numeric data, and it is optional, as sometimes some data is critical and any form of compression is not acceptable. And in these cases, we just store the original data and use the lossless compression methods.

4.3 Relationship with Smart Contracts

Currently our system does not contain a smart contract component. In our system design, the interaction behaviors between the blockchain and components, such as writing the records, are maintained by the miner nodes. This is because we mainly focus on the medical applications, and the direct approach helps to improve the system efficiency. In fact our system can be either built on dedicated miners or on general blockchain systems. So this interaction mechanism can also be implemented by a general blockchain system (such as the Ethereum) with the corresponding smart contracts, and it is easy to translate the protocol described in our paper to the corresponding smart contracts. In our future work, we will also consider establishing our framework on a universal blockchain and translate the protocols to smart contracts, as it is easier to promote the system based on a universal blockchain.

4.4 Protocols

Here we give the detailed descriptions of the core protocols.

The user updates the collected data to the EDH. The EDH maintains an off-blockchain key-value store. A customer requests a service to provide the health data analysis for a user. The service then finds the corresponding EDH, and requests for data or SCP execution based on the EDH's local policy, fully controlled by the user. The EDH node will decide the permission based on its local policy controlled by the user, with information of the service, e.g. the credibility of the service recorded on the blockchain. If the check passes, the EDH will return the data or the SCP execution result to the service. The details of the interactions between the service and the EDH is given in protocol 1. After retrieving the required data from the EDH, the service can get the analysis result and then sends it to the customer, and finishes the whole transaction. After the whole process, all the participants can score each other. The average of the credibility the service received in a time period, such as a month, could be regarded as the credibility of the service. If the EDH wants to check the credibility of one service, it can query the credibility scores recorded on the blockchain.

Some nodes in the system not only participate in data sharing, but also append blocks to maintain the ledger, on the principle of voluntariness, as blockchain 'miners'. These nodes apply the PoW consensus to compete to solve a cryptographic puzzle for the ownership of new blocks [13, 27]. The latest blockchain status is then broadcasted to the network by the winning node. Other nodes then update their status accordingly.

A typical system flow is shown in Figure 5 and the corresponding description of each step is illustrated as

follows:

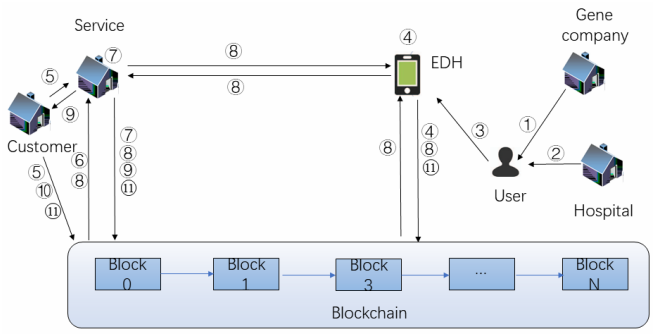

Figure 5. A case demonstrating the flow of the system

(1) The user collects the data from a gene company.

(2) The user collects the data from a hospital.

(3) The user updates the data to the EDH.

(4) The EDH compresses the data and updates the key-value store for the data. Then it writes the corresponding data hash record on the blockchain.

(5) A customer requests a service to provide the health data analysis for a user. It first writes the corresponding REQ_SERVICE transaction record (record A) with data hash of the request content, and then sends the request to the service.

(6) The service receives the request. It first verifies the REQ_SERVICE transaction record, and finds the user's EDH from the blockchain.

(7) To begin the analysis, the service writes a BEGIN_SERVICE transaction record (record B, with PTRI pointing to record A).

(8) The service begins the data request process with the EDH, with the corresponding transaction records $C_{1,a}, C_{1,b}, C_{1,c}, \ldots, C_{n,a}, C_{n,b}, C_{n,c}$, where n denotes the number of interactions, See Protocol 1 for details.

(9) Based on the data from the EDH (either directly returned data or the SCP execution result), the service gets the final analysis result, and return it to the customer. It also writes a COMPLETE_SERVICE transaction record D (with PTRI pointing to record $C_{n,b}$).

(10) The customer receives the analysis result and writes the ACK_COMPLETE_SERVICE transaction record E (with PTRI pointing to record D) to acknowledge its reception.

(11) All the participants in the process can write the credibility score record for each other. For example, the customer can evaluate the service's quality based on the analysis result, and the service can evaluate the EDH's quality based on the acquired data.

Protocol 1. Process that a service connects to an EDH and asks for some data

1. The service connects to an EDH and setup a secure tunnel.
2. The service shows its public key S_{vk}.
3. The EDH sends a random string T to service.
4. The service sends the signature S of T with its private key S_{sk}.
5. The EDH verifies T with public key S_{vk}, and closes the tunnel if it is incorrect.
6. The EDH checks the legality of the service, e.g. checks the credibility of the service from the blockchain, and evaluates the legality according to its local policy. If the check fails, it will close the tunnel.
7. The EDH determines the interaction mode, either the direct data acquisition or it needs an SCP execution (see section 4.1.3).
8. Run the data acquisition loop using protocol 2 for the former case and protocol 3 for the later one.

Protocol 2. Direct data acquisition

Performs the loop until service closes the tunnel.

(a) The service determines the required keys, and writes the REQ_DATA transaction record $C_{i,a}$ with the hash of the required keys of the data acquisition, and PTRI pointing to the previous one during the whole process (similar for the later PTRIs). Note that i is the loop counter ranges from 1 to n (the total loop steps).

(b) The service sends a request to the EDH including the required keys.

(c) The EDH receives the request, verifies it from the blockchain, checks the local key permissions controlled by the user, returns the query results (return null if the check fails), and writes the RET_DATA transaction record $C_{i,b}$.

(d) The service receives the returned data, verifies the data using the hash on the blockchain, and writes the ACK_DATA transaction record $C_{i,c}$. The process will end if the verification fails.

Protocol 3. SCP data acquisition

Performs the loop until service closes the tunnel.

(a) The service determines the required output keys for the SCP program, and writes the REQ_SCP transaction record $C_{i,a}$ with the hash of the required output keys of the SCP execution, and PTRI pointing to the previous one during the whole process (similar for the later PTRIs). Note that i is the loop counter ranges from 1 to n (the total loop steps).

(b) The service sends a request to the EDH including the scp program and its output keys.

(c) The EDH receives the request, verifies it from the blockchain, checks the local key permissions (to verify if the SCP can return the output keys), executes the SCP program, fetches and verifies its output key-value pairs, writes the RET_SCP

transaction record $C_{i,b}$, and sends the output back to the service. If any exception occurs during the execution, the EDH will record and return the error immediately. A RET_SCP transaction record $C_{i,b}$ with the error will also be recorded.

(d) The service receives the returned data or error, verifies it and writes the ACK_SCP transaction record $C_{i,c}$. The process will end if the verification fails.

Here we give more details on the SCP execution (step c) of Protocol 3. Recall that each user holds a private/public key pair (U_{sk}, U_{vk}). The user data stored in the EDH can be viewed as a collection of key-value pairs, and a value can be indexed uniquely by its key.

We first list utility functions in Protocol 4, and then the execution in Protocol 5.

Protocol 4. Utility functions

1. check_access_valid (S_{vk}, key): bool // An EDH specific function which determines whether the service S_{vk} has the right to access the key.
2. find_keys (S_{vk}, key_pattern: Regex): KeyCollection // It returns all keys satisfying key_pattern, and with the property check_access_valid (S_{vk}, .) = true. The Regex represents the regular expression class.
3. access_data (S_{vk}, key): AccessDataResult // If service S_{vk} has the right to access the key, it returns the corresponding value, else it throws an exception. The AccessDataResult is the data class for the value result.
4. execute_code (S_{vk}, code): ExecutionResult // A code execution function that executes the code in a restricted environment, with the ability to access the granted EDH data on the SCP. The code will execute, access permitted data, analyze it, and return the result as the ExecutionResult data class. For simplicity, currently we restrict the code to be the JavaScript code, which will be compiled to byte codes before execution. The two functions the code can access are find_keys and access_data, with the S_{vk} parameter fixed. The returned ExecutionResult must be an array of key-value pairs, verified by the EDH to protect privacy.

Protocol 5. SCP execution

1. Get S_{vk} from the communication channel.
2. if (!check_access_valid(S_{vk}, EXECUTE_CODE)) throw Error ("Service cannot execute code"); // EXECUTE_CODE is a special key used to indicate whether a service can execute code.
3. return execute_code (S_{vk}, code). //The code is provided by the service node for analyzing data.

4.5 Complexity Analysis

We analyze the EDH execution time complexity in our system, including the process of the data query and the code execution. For the access_data function in Protocol 4, we prebuild a hash table in the EDH for the key/value store, and the complexity of such an operation is $O(1)$. For the regular expression search among the keys, we use the compressed generalized suffix tree structure, which takes $O(n)$ time to construct, where n is the sum of all the key lengths [28]. It costs $O(m)$ time for inserting or deleting a key with length m. For a substring query (regular expression in the form $.*S.*$, where S is a normal substring to find), it takes $O(m+occ)$ time [28], where $m = lenth(S)$ and occ is the number of occurrences. The second scenario is the range query, e.g. get all the keys in range (in string alphabet order), the complexity can also be described by $O(m+occ)$, where m is the sum of lengths of the two strings and occ is the number of keys in the range. This kind of query is useful for filter keys according to categories, e.g. medical records in hospital A from 2019.10.1 to 2019.10.31 are described by range (/EHR/hospitalA/20191001, /EHR/hospitalA/20191031). For the wildcard queries, we can split the string by the * and ? characters, and query each part using the suffix tree, and then merge them. The overall complexity is $O(m+occ')$ where m is the wildcard query length and occ' is the sum for all the located candidates for each part. There is also a faster wildcard searching algorithm with the complexity $O(m+occ)$, where occ is the total number of occurrences [29], but it has a higher space complexity. For general regular expressions the complexity is sublinear in n [28]. And there are also fast algorithms for other subclasses of regular expressions, e.g. [30]. In general, in most of our situations where the queries are limited to the substring/range queries and the query length m is bounded, the complexity is approximately $O(occ)$.

The overall running time complexity is $O(L_C) + q * N_Q$, where L_C is the count of executed byte code instructions, q is the number of key queries using regular expressions, and N_Q is the average time complexity for each query as described above. Generally speaking, $q * N_Q$ is proportional to the number of returned keys in most cases, which is often bounded by the number of executed byte code instructions, as generally the code will traverse among the returned keys. The overall time complexity can be considered as $O(L_C)$ for most scenarios, which is

quite acceptable.

4.6 Incentive Mechanism

Medical stakeholders are incentivized to participate in the system by the advantages of the system. And the users can store information in a decentralized system to take full control of the data. Meanwhile, once the data including the hash value, the transactions and so on are stored on the blockchain, they are immutable and persistent. This property enables the system to store evidence for the whole activity history.

Furthermore, we motivate medical stakeholders, such as researchers and so on, to actively participate in the maintenance of the ledger. The first incentive mechanism is the credit award, the ledger maintainers can get some credibility scores in a bitcoin-like mechanism. The credibility is important in most of the data transactions, and the participants need certain amount of credibility to retrieve the data, so this incentive is important and quite attractive. The second incentive mechanism is the direct data award similar to the bounty query in [9]. When an EDH finishes writing the RET_DATA (return data) transaction record, it may also write a reward data transaction record including the reward query (e.g. the average weight of people during a period on the same EDH server, and the data privacy can be controlled by the users) and the result hash. When the block including the transaction is later mined, the miner then gets the reward. It then sends a query to the EDH and the EDH can identify the miner from the blockchain (the miner public key will be kept in the block), and then returns the data to it.

5 Conclusion

Here we introduce a new decentralized system for private medical data sharing and access control. Our system enables not only data sharing between EDH and medical institutions while retaining data privacy, but also the reduction of storage space. To respect data privacy, SCP is used to run the executable code provided by the service in the EDH without private data leakage. The hash values of data are stored on the blockchain to ensure that accessed data is tamper proof. To reduce storage space, we utilize an off-blockchain storage method leveraging the data sparsity. Since the blockchain ledger is tamper proof, it keeps an auditable history of the medical data accesses.

The system we propose is comprehensive, accessible and credible. Our system can not only effectively prevent data leakage by SCP when the data analysis request is permitted, but also reduce off-blockchain storage while maintaining as much medical data information as possible. In our system, meaning records with predefined schemas is proposed, which can normalize data format stored on the blockchain. Our system can save the storage cost, which makes it feasible to collect the massive medical data. Taking the key properties of decentralization, our system does not rely on a centralized entity but lots of participating entities which can avoid the failure of a single point. Through our system, companies can access large amounts of data to provide personalized services. Researchers can access large-scale medical data, which will help discover wide-reaching patterns in order to achieve progress in precision medicine.

As in this paper we mainly focus on the high-level system design, perhaps the most straightforward future work is to provide an efficient implementation for the proposed system and deploy it in the real world. As we use a public blockchain, the underlying mechanism is very similar to the bitcoin blockchain, which provides a good reference for the implementation. The blockchain related part of the proposed protocols should be verified and guaranteed directly by the blockchain miners. Another possible future work is to rely on a universal blockchain with smart contract support, as mentioned in the previous section. Improving credibility evaluation is also a potential research direction. The disadvantage of our system is the lack of intelligent analysis of credibility. With the development of machine learning, we can further improve the evaluation with machine learning methods.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant no. U1736210, 31501081, and in part by the Next-generation Platform of Data Sharing and Privacy Protection of Tsinghua-Fuzhou Data Technology Research Institute under Grant no. TFIDT2018004.

References

[1] G. Zyskind, O. Nathan, A. Pentland, Decentralizing Privacy: Using Blockchain to Protect Personal Data, *2015 IEEE Security and Privacy Workshops (Spw)*, San Jose, CA, USA, 2015, pp. 180-184.

[2] Personal Data: The Emergence of a New Asset Class, *An Initiative of the World Economic Forum*, January, 2011, http://www3.weforum.org/docs/WEF_ITTC_PersonalDataNewAsset_Report_2011.pdf.

[3] E. J. Topol, Individualized Medicine from Prewomb to Tomb, *Cell*, Vol. 157, No. 1, pp. 241-253, March, 2014.

[4] L. J. Kish, E. J. Topol, Unpatients-Why Patients Should Own Their Medical Data, *Nature Biotechnology*, Vol. 33, No. 9, pp. 921-924, September, 2015.

[5] U. S. D. o. HHS, Report on Health Information Blocking, *REPORT TO CONGRESS*, April, 2015, https://www.healthit.gov/sites/default/files/reports/info_blocking_040915.pdf.

[6] M. P. Andersen, J. Kolb, K. Chen, G. Fierro, D. E. Culler, R. A. Popa, *WAVE: A Decentralized Authorization System for*

IoT via Blockchain Smart Contracts, Technical Report No. UCB/EECS-2017-234, December, 2017.

[7] J. Evans, Bitcoin 2.0: Sidechains and Ethereum and Zerocash, oh My!, https://techcrunch.com/2014/10/25/bitcoin-2-0-sidechains-and-zerocash-and-ethereum-oh-my/, 2014.

[8] S. Nakamoto, Bitcoin: A Peer-to-peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, 2008.

[9] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, MedRec: Using Blockchain for Medical Data Access and Permission Management, *Proceedings 2016 2nd International Conference on Open and Big Data - Obd 2016*, Vienna, Austria, 2016, pp. 25-30.

[10] Q. Xia, E. B. Sifah, K. O. Asamoah, J. B. Gao, X. J. Du, M. Guizani, MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain, *IEEE Access*, Vol. 5, pp. 14757-14767, July, 2017.

[11] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, F. Wang, Secure and Trustable Electronic Medical Records Sharing Using Blockchain, *AMIA Annual Symposium Proceedings Archive*, Vol. 2017, pp. 650-659, April, 2018.

[12] F. Tschorsch, B. Scheuermann, Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies, *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 3, pp. 2084-2123, Third Quarter, 2016.

[13] M. Jakobsson, A. Juels, Proofs of Work and Bread Pudding Protocols, *IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, Leuven, Belgium, 1999, pp. 258-272.

[14] B. C. Neuman, T. Ts'o, Kerberos - An Authentication Service for Computer Networks, *IEEE Communications Magazine*, Vol. 32, No. 9, pp. 33-38, September, 1994.

[15] K. Zeilenga, *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*, RFC 4510, June, 2006.

[16] Z. Yang, K. Yang, L. Lei, K. Zheng, V. Leung, Blockchain-based Decentralized Trust Management in Vehicular Networks, *IEEE Internet of Things*, Vol. 6, No. 2, pp. 1495-1505, April, 2019.

[17] G. Caronni, Walking the Web of Trust, *IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Gaithersburg, MD, USA, 2000, pp. 153-158.

[18] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and. Braynard, Networking Named Content, *Communications of the ACM*, Vol. 55, No. 1, pp. 117-124, January, 2012.

[19] M. Marwan, A. Kartit, H. Ouahmane, A Cloud Based Solution for Collaborative and Secure Sharing of Medical Data, *International Journal of Enterprise Information Systems (IJEIS)*, Vol. 14, No. 3, pp. 128-145, July-September, 2018.

[20] Y. A. d. Montjoye, E. Shmueli, S. S. Wang, A. S. Pentland, openPDS: Protecting the Privacy of Metadata through SafeAnswers, *Plos One*, Vol. 9, No. 7, e98790, July, 2014.

[21] K. Fan, S. Y. Wang, Y. H. Ren, H. Li, Y. T. Yang, MedBlock: Efficient and Secure Medical Data Sharing via Blockchain, *Journal of Medical Systems*, Vol. 42, No. 8, Article number: 136, August, 2018.

[22] H. Yin, D. C. Guo, K. Wang, Z. X. Jiang, Y. Q. Lyu, J. Xing, Hyperconnected Network: A Decentralized Trusted Computing and Networking Paradigm, *IEEE Network*, Vol. 32, No. 1, pp. 112-117, January-February, 2018.

[23] S. P. Wang, Y. L. Zhang, Y. L. Zhang, A Blockchain-Based Framework for Data Sharing with Fine-grained Access Control in Decentralized Storage Systems, *IEEE Access*, Vol. 6, pp. 38437-38450, June, 2018.

[24] Z. B. Zheng, S. A. Xie, H. N. Dai, X. P. Chen, H. M. Wang, Blockchain Challenges and Opportunities: A Survey, *International Journal of Web and Grid Services*, Vol. 14, No. 4, pp. 352-375, October, 2018.

[25] A. Ng, *Sparse Autoencoder*, https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf, 2011.

[26] J. Wright, A. Ganesh, S. Rao, Y. Peng, Y. Ma, Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices by Convex Optimization, *Neural Information Processing Systems*, Vancouver, Canada, 2009, pp. 1-9.

[27] D. Qin, C. X. Wang, Y. M. Jiang, RPchain: A Blockchain-Based Academic Social Networking Service for Credible Reputation Building, *International Conference on Blockchain*, Seattle, WA, USA, 2018, pp. 183-198.

[28] P. Weiner, Linear Pattern Matching Algorithms, *14th Annual Symposium on Switching and Automata Theory*, Iowa City, Iowa, USA, 1973, pp. 1-11.

[29] P. Bille, I. L. Gørtz, H. W. Vildhøj, S. Vind, String Indexing for Patterns with Wildcards, in: F. V. Fomin, P. Kaski (Eds.), *Algorithm Theory – SWAT 2012. SWAT 2012. Lecture Notes in Computer Science*, Vol. 7357, Springer, Berlin, Heidelberg, 2012, pp. 283-294.

[30] R. A. Baeza-Yates, G. H. Gonnet, Fast Text Searching for Regular Expressions or Automaton Searching on Tries, *Journal of the ACM*, Vol. 43, No. 6, pp. 915-936, November, 1996.

Biographies

Qingzhu Yang received the Ph.D. degree from University of Chinese Academy of Sciences in 2013. She is an assistant researcher at the Department of Automation, Tsinghua University. Her research interest covers statistics analysis, machine learning and bioinformatics.

Qiao Liu received the B.E. degree in ShenYuan Honors College, Beihang University, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University, China. He joined the MOE Key laboratory of Bioinformatics since 2016. His research interests

include applied machine learning, biomed and bioinformatics.

Hairong Lv received the Ph.D. degree from Tsinghua University in 2007. He is currently an associate researcher at the Department of Automation, Tsinghua University. His research interest covers artificial intelligence and blockchain. He is corresponding author of this paper.