IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2017, accepted July 11, 2017, date of publication July 24, 2017, date of current version August 14, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2730843

MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain

QI XIA[1], EMMANUEL BOATENG SIFAH[2], KWAME OMONO ASAMOAH[2], JIANBIN GAO[3], XIAOJIANG DU[4], (Senior Member, IEEE), AND MOHSEN GUIZANI[5], (Fellow, IEEE)

[1]Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[3]Center for Digital Health, School of Resource and Environment, University of Electronic Science and Technology of China, Chengdu 611731, China
[4]Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA
[5]Electrical and Computer Engineering Department, University of Idaho, Moscow, ID 83844 USA

Corresponding author: Qi Xia (xiaqi@uestc.edu.cn)

This work was supported in part by the applied basic research programs of Sichuan Province under Grant 2015JY0043, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2015J154, Grant ZYGX2016J152, and Grant ZYGX2016J170, in part by the programs of International Science and Technology Cooperation and Exchange of Sichuan Province under Grant 2017HH0028, in part by the Key Research and Development Projects of High and New Technology Development and Industrialization of Sichuan Province under Grant 2017GZ0007, in part by the National Key Project in Cyber Security under Grant 2016QY04W0800,02,03, and in part by the National Key Research and Development Program of China under Grant 2016QY04W0800,02,03.

ABSTRACT The dissemination of patients' medical records results in diverse risks to patients' privacy as malicious activities on these records cause severe damage to the reputation, finances, and so on of all parties related directly or indirectly to the data. Current methods to effectively manage and protect medical records have been proved to be insufficient. In this paper, we propose MeDShare, a system that addresses the issue of medical data sharing among medical big data custodians in a trust-less environment. The system is blockchain-based and provides data provenance, auditing, and control for shared medical data in cloud repositories among big data entities. MeDShare monitors entities that access data for malicious use from a data custodian system. In MeDShare, data transitions and sharing from one entity to the other, along with all actions performed on the MeDShare system, are recorded in a tamper-proof manner. The design employs smart contracts and an access control mechanism to effectively track the behavior of the data and revoke access to offending entities on detection of violation of permissions on data. The performance of MeDShare is comparable to current cutting edge solutions to data sharing among cloud service providers. By implementing MeDShare, cloud service providers and other data guardians will be able to achieve data provenance and auditing while sharing medical data with entities such as research and medical institutions with minimal risk to data privacy.

INDEX TERMS Access control, blockchain, cloud computing, data sharing, electronic medical records, privacy.

I. INTRODUCTION

In modern societies, cultures and organized groups, the dissemination of medical data has been perceived to be a breakthrough for the discovery of new techniques and therapies for curing diseases [1]. The key drive for the above mentioned statement is due to the digitization, electronic storage and remote accessibility of medical data by professionals [2]. These records are generated by hospitals after patient visits thereby making patients sole owners of electronic medical records.

With the advent of the technology age and the subsequent collection of huge volumes of data that have ushered in the big data era, sharing data offers attractive value on prospects of which are still uncovering [3]. The importance of data and the value inherent in its dissemination has given birth to business entities that collect, process, analyze, store, and given the appropriate incentive sharing of data with other interested parties [4], [5]. This has spawned the interest of several industries with a focus on cloud storage and processing mechanisms, data analytics and data provenance rendering traditional industries dependency on data availability on their operations and survival [6], [7]. In achieving the high demands on Big Data storage, several stakeholders have resorted to cloud storage and computing to provide suitable

solutions to pressing storage and processing demands [8], [9]. The increased popularity of cloud services has drawn the interest of users which span from patients, medical institutions and research institution to big cooperation's to store their acquired data on cloud repositories [10], [11]. Cloud service providers are mandated to however provide a controlled, cross-domain and flexible data sharing of medical data stored in their repositories to beneficiaries [12].

However, cloud service providers (CSP) struggle with a lack of collaboration for sharing data due to the adverse risks posed on exposing the contents on their data [13]. Conclusions can be drawn from obvious facts that, healthcare parties (institutions and policy makers) go as far to disallow data sharing while blocking protocols that facilitate data dissemination [14]. For data owners and custodians, there is an existing risk of collected data being vulnerable in the hands of malicious data users. For such risks, policy makers lessen intent of exposing data content by instilling policies that exploit the fear of data users. Although policies work in the favor of data owners and custodians, the fear of breaching regulations and the ensuing penalties to be paid in both financial and reputation terms foster an atmosphere of distrust which ensures data sharing does not occur [15].

For establishing right inducements geared towards data sharing while highlighting attractive features of such acts, there remains the issue of loss of control over the data [16]. Traditionally, once the data leaves the custodians system where the data was first collected or generated, there is no control over what the next user can do with it [17]. This permits malicious users to abuse data, causing data owners and custodians several legal and reputation-attacking problems with industry regulators.

Several cryptographic methods have been proposed to address these problems arising from sharing of medical data but have still been insufficient [18]–[21]. Nonetheless, the blockchain is seen as a strong fit to provide a suitable solution to addressing this problem through its attractive features such as immutability and decentralization [22]–[26].

In this paper, we proposed a blockchain based solution for sharing medical data among cloud service providers while providing data access control, provenance and auditing. Actions of data beneficiaries are constantly monitored through mechanisms mentioned later in the paper and breaches are addressed accordingly by revoking access to the data.

II. RELATED WORKS

In this section, research trends pertaining to data via cloud service providers and access control with emphasis on the improving blockchain technology are outlined. Sundareswaran *et al.* [27] proposed ensuring distributed accountability for data sharing in the cloud, an approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Their approach allows a data owner to audit content as well as enforce strong backend protection when required.

Zyskind *et al.* [28] used the blockchain for access control management and for audit log security purposes, as a tamper proof log of events. Enigma is a proposed decentralized computation platform based on an optimized version of secure multi-party computation (sMPC). The different parties altogether store and run computations on data while keeping the data completely private.

Xia *et al.* [29] proposed a blockchain-based data sharing framework that sufficiently addresses the access control challenges associated with sensitive data stored in the cloud using immutability and built-in autonomy properties of the blockchain. They employed the use of secure cryptographic techniques to ensure efficient access control to sensitive shared data pool(s) using a permissioned blockchain and design a blockchain-based data-sharing scheme that permits data users/owners to access electronic medical records from a shared repository after their identities and cryptographic keys have been verified. The requests after verification and onward servicing form part of a closed, permissioned blockchain.

Ferdous *et al.* [30] presented DRAMS, a blockchain-based decentralized monitoring infrastructure for a distributed access control system. The main motivation of DRAMS is to deploy a decentralized architecture which can detect policy violations in a distributed access control system under the assumption of a well-defined threat model.

Blockchain-based access control management was described in more detail in Thomas and Alex's system. The ChainAnchor system provides anonymous but verifiable identity to entities trying to perform transactions to a permissioned blockchain. The Enhanced Privacy ID (EPID) zero-knowledge proof scheme is used to achieve and prove the participants anonymity and membership [26].

Hassan *et al.* [31] introduced a hybrid network model for efficient real-time WBAN media data transmission. The proposed network architecture combines WBAN and Cloud for valid data sharing and delivery. It also utilizes the combination of CCN or NDN with adaptive streaming technique to support uninterrupted media healthcare content delivery to multiple patients and physicians, as well as help to reduce packet loss.

In this work, we provide a secured blockchain-based data sharing of electronic medical records among untrusted parties. The main contribution of our work is to provide data provenance, auditing and secured data trailing on medical data. The various literatures reviewed in this section provides insufficient mechanisms in achieving data provenance, auditing and data trailing on medical data. It should be mentioned that our system relies on smart contracts to effectively monitor the behavior of data out of the custodians care.

III. PRELIMINARIES

In this section, we formally define the preliminaries used in our blockchain-based data sharing system among untrusted parties. We highlight the blockchain network with side-blocks as part of individual components in addition to triggers put in place to achieve the system. A brief outline on the

supposed behavior of cryptographic behavior required for the system is further described.

A. BLOCKCHAIN NETWORK

The blockchain is a distributed database that contains an ordered list of records linked together through chains, on blocks. Blocks can be defined as individual components that contain information relating to a particular transaction. An example of such information can be a log on a single event (requestor needing data from the system). A blockchain network maintains a continuous growing list of records which are immutable. Due to this reason, many systems built on the blockchain technology achieve secured distribution of assets among untrusted clients.

For our data sharing among untrusted parties' system, the processing and consensus nodes are entirely responsible for broadcasting blocks into the blockchain network after processing requests. Forms generated per request are developed into blocks and later broadcast during the process of delivering package to a requestor. This action culminates the completion of a block and permits the broadcast of the block into the blockchain network. There are multiple threads of blockchains in the network, identified uniquely using a requestor's identity. Threading blocks to their parent blocks are used to keep a contiguous log of well-ordered logs developed from requests by different requestors. In structuring the network this way, we point to the fact that each block in a particular string represents different instances of request made. These are indexed and updates by the smart contracts in a particular child-block are appended to the parent block as a side-block. The importance of implementing side-blocks is to maintain an effective log and efficient fetching of blocks with instances on questioning and investigation for an occurrence of breach of terms by the data owner to requestor of the data.

Side-blocks appended to their parent blocks contain reports from the smart contract reports that are indexed thereby maintaining accuracy on identical reports with violations stored concurrently in a smart contract permissions database. The structure of creating multiple threads for our blockchain network and creating side-blocks on parent blocks, aggregates to a comprehensive collection of reports.

As mentioned earlier, a block is developed from a form created from a request. An entire block is made up of a single request which spans from when it was made till when the package is ready for delivery. The processing and consensus node is responsible for the maintenance of the blocks and blockchain network. As a matter of fact, processing and consensus nodes are the only entities with direct access to the blockchain network. Whilst monitoring the blocks with side-blocks, nodes alert the system on breaches to the use of data.

B. CRYPTOGRAPHIC KEYS

Cryptographic keys indicate the sets of keys highlighted to execute specified tasks relating to the security of a framework. For the blockchain-based data sharing scheme, we highlight keys needed to achieve confidentiality for transactions between systems via untrusted channels. These keys help to guarantee for a level of security of the scheme.

For our system, there is the need to adopt cryptographic primitives for secured transfer of query results between untrusted data sharing parties to prohibit the influence of malicious individuals and eavesdroppers on a transaction between the sharing parties. A description on the primitives and keys that would best suit the system are enlightened. The keys include:

- Requestor private key, generated by a requestor and digitally used to sign requests for data access.
- Requestor public key, generated by a requestor and sent to data owners authenticator to be used as verification of requestors identity for data access. The requestors public key is also used to encrypt a package to be sent to the requestor by the authenticator.
- Authenticator contract key, key pair generated by authenticator and attached to smart contract in a package used for encrypting reports that leave the requestor's system to data owner's system.

For a requestor who wants access to sets of records from a data owner, the requestor generates a key pair (Requestor private and public keys), stores the private key and shares the public key with the data owner or other data owner the requestor might access data from in the future. The requestor creates, sign using the requestor private key and send a request to a data owner. On reception, the data owner confirms the request by verifying the signature with the requestor public key.

Results on computations on data requested are tagged on the data and further processed by data owner by appropriate entities in the data owner's system. The data owner encrypts the whole package (final processed data to be delivered to requestor) with an authenticator contract key and shares the package with the requestor. On reception, the requestor decrypts the encrypted package and uses the data.

To adequately secure the transfer of actions performed on the data from the requestor to the data owner, the reporting of actions generated from the cryptographic functions flagged on the data should be encrypted using the authenticator contract key tagged to the contracts generated from the data originator (owner) which would be sent to a database a secured database.

C. TRIGGERS

Triggers are entities that connect processes between the query structure and the blockchain environment. The key importance of implementing triggers for our system is to enable smart contracts to directly connect from outside the system to our system since smart contracts cannot directly interact with structures outside of its environment, thus the blockchain network. Triggers hold no information and just act as an intermediary between the query layer and the data structuring and provenance layer of the system. The trigger incorporates interfaces internally and externally and updates the process

states to and from the query system by the inbound and outbound smart contracts based on external and internal features of the contract.

IV. DESIGN FORMULATION

A. SYSTEM MODEL

We formulate data sharing mechanism used by the blockchain-based data sharing among untrusted parties for data security and provenance. Categories of structures of the system are displayed in ***Figure 1*** and grouped into four main layers namely:

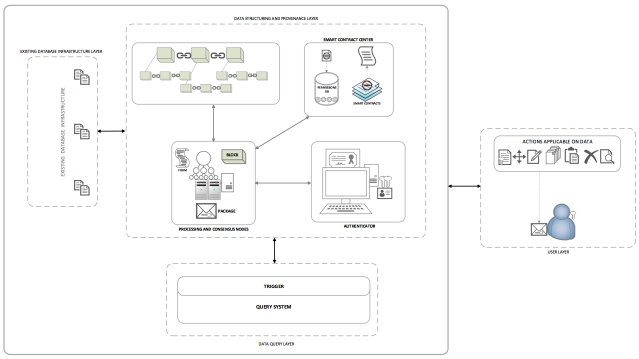

FIGURE 1. System design with four main layers and individual components.

1) User layer: The user layer consists of all the different classifications of users whose intentions are to access data from the system for research or other useful purposes. The intent of most users is to help analyse data for research purposes. Examples of users can be healthcare organizations such as hospitals, research institutions as well as universities, individual research personnel's and governmental bodies.
2) Data Query layer: The data query layer consists of sets of querying structures that access, process, forward or respond to queries posed on the system. Queries on the systems may be requests to access data from the existing database infrastructure. The data query layer directly interfaces with the data structuring and provenance layer and has mechanisms, implemented to interpret and translate actions between the data structuring and provenance layer and the outside environment (out of the system). Users directly interact with the query layer for data requests. Components of the data query layer are;
   - Querying system: The query system is responsible for processing request into a format desired by the data structuring and provenance layer. Requestors send a query which is processed by the query system whose output is a value (data requested) for the appropriate request. The final role of the query system is to send a response based on the request to the requestor.
   - Trigger: The trigger is responsible for translating actions to and from the smart-contract environment since smart contracts cannot fully function outside the blockchain environment.
3) Data Structuring and Provenance layer: Data structuring and provenance layer consist of individual components that help process request for access to data from the existing database infrastructure layer. The layer additionally performs computations on requested data and tag data with functionalities that monitor every action performed on the data. Algorithms and structures are implemented in the data structuring and provenance layer to report actions which are securely stored in a database and indexed appropriately triggering an action on the monitored data if need be. Results of every action completed are broadcast into an immutable network to guarantee trust-less and fair auditing. Finally, the layer has a responsibility of authenticating every request and actions pertaining to data access from the entire system. The different entities in the data structuring and provenance layer;
   - Authenticator: The authenticator is responsible for verifying the legitimacy of requests sent by requestors to the data owners system. The authenticator generates authenticator contract keys that are used to encrypt actions from the data in the user's environment to the data owner's system. The authenticator tags the encryption keys to the entity responsible for reporting such actions. Additionally, the authenticator encrypts a package which contains the data requested by the user and is finally delivered to the appropriate requestor.
   - Processing and consensus nodes: The processing and consensus nodes process forms created for requests which are later developed into blocks and broadcast into the blockchain network. In addition, the consensus node is tasked to create packages containing the requested data and smart contract to be delivered to the requestor.
   - Smart Contracts: Smart contracts are designed functions that are activated and executed on the reception of an action. The smart contracts generated are embedded with cryptographic keys enabling contracts to encrypt reports generated from the activation of actions. The main role of a smart contract is to identify actions performed on the sent data and report the data into a database. Finally, smart contracts revoke access to the violated data with actions not permissioned on the data by the requestor.
   - Smart Contract Permissioned Database: The smart contract permissioned database is a report violation storage and action entity which is activated on the reception of a violation report by the processing and consensus nodes for all reports received by the smart contracts from the requestor. Lists of actions to be carried out by the data owner on violation of contracts are stored in the smart contract

permissions database. Receipts are kept for each action activated to adequately provide consistency for accountability for data forensic and auditing for future reference.

- Blockchain Network: The blockchain network is composed of individual block broadcast into a network and chained together in a chronological method. The main role of the blockchain network is to maintain a chronologically distributed immutable database of actions on the delivery and request of data from the system. The blockchain also maintains a side-block for individual actions pertaining to specific data reported by the smart contracts into the data owner's system.

4) Existing Database Infrastructure layer: The existing database infrastructure layer consists of already established database systems implemented by individual parties to accomplish specific tasks. Such database systems are only accessible by authorized personnel of such companies since they house sensitive information which requires secure mechanisms to adequately protect such sensitive data. For access to data from such databases, requested datasets are passed through sets of computations to desensitize the data before they are shared.

B. THREAT MODEL AND GOALS

Our threat model is categorized under two levels:

- Data threat level: Data threat level defines violations from actions entities can perform on data without knowledge of the data owner thereby risking the privacy and data value of the received by a data user. Such users' compromise privacy mechanisms imposed in a given data by desensitizing the acquired data.
- Report swap threat level: A report swap threat level defines malicious users whose intents are to acquire data and abuse access rights whilst altering or swapping reports generated by smart contracts tagged to the data. Such users' compromise value and privacy for a requested data.

We aim to achieve the following threat model goals:

- Ensure secure data provenance and auditing by implementing smart contracts which closely monitor all actions performed on a data thereby exposing the data threat level for a given malicious user whilst applying access revoking methods.
- Ensure secured confidentiality of reports by self retrieving keys attached to smart contracts for encryption, limiting the actions of malicious users on smart contract reports.

V. CONSTRUCTION OF MEDSHARE

In this section, we present a construction of entities and functions of components necessary for the secured sharing of data among untrusted parties. We outline designed structures that realizes data sharing by presenting our data access system which aims to provide a suitable sharing scheme whilst preserving the required security properties of the blockchain.

A. DESIGN APPROACH

- System setup:
  A user sends a request for data access to a system. The data request is signed by the user using a ''pre-generated'' requestor private key. First of all, the request is received by the query system an entry point for recording and processing requests. The query system forwards the request to the data structuring and provenance layer by the triggers since triggers translate the query (request) into a structure that can be understood by the data structuring and provenance layer. The authenticator receives the request and verifies the legitimacy by verifying the signature using the requestors public key which was generated and shared before the request was sent. If the signature is valid, the process is accepted else dropped for invalid requests.
- Request file:
  For a valid request, the authenticator forwards the request to the processing and consensus nodes where processing of requests into forms are completed. The form generated contains a hash of the timestamp of when the request was received and a hash of the ID of the requestor. The purpose for the data request is tagged to the form and then forwarded to the existing database infrastructure. The existing database infrastructure receives the form, retrieves the data and sends the retrieved data to the data structuring and provenance layer. This is received by the processing and consensus node and a hash of this timestamp is concatenated to the existing record of a hashed timestamp for the request. The processing and consensus nodes send a request to the smart contract center for appending sets of rules to the requested data. A smart contract is generated and tagged to the form which contains the data indexed to form some sort of adjacency with the related block.
- Package delivery:
  Results of the processed data are then sent to the authenticator to generate an authenticator contract key and tag the encryption key of this generated set to the smart contract on the data. The importance of tagging the smart contracts to the data is to ensure data auditing and traceability is achieved. The processing and consensus node process a block based on the information relating to the request and broadcasts the block into a blockchain network. The block forms a part of already existing blocks in relations to the requestor and is tagged with an identity to uniquely identify the different blocks in the network. This is achieved through chronology and perfect ordering. The data is encrypted, forming a package encrypted, and tagged with the ID of the requestor. The result is forwarded to the query system through triggers and is finally distributed to respective requestors.

Package refers to a completely processed data file along with certain parameters intended for the requestor. A package contains a data ID, payload (data) and a smart contract. A package is created from processing on received data from the existing database infrastructure by the consensus nodes. Monitoring of the lifeline of a package is made effective by attaching a smart contract to the package. The completion of a package is culminated by the authenticator by encrypting the package into a format which can only be readable by a valid and appropriate requestor with the right private key.

- Auditing and provenance:
  The requestor receives the package and decrypts the encrypted file using the requestor private key. To ensure the efficiency of every computation tagged to the data (smart contracts), decryption of the data should activate the smart contract to retrieve the encryption key tagged to itself and encrypt an action of **decrypted data** which is sent to the query system to appropriately forward the report to the data structuring layer to the processing and consensus node. The processing and consensus node processes the report into a side-block and appends the result to the parent block pertaining to that particular data request as part of traces on actions for the data. The received report is sent to activate violation actions that can be applied to the received data from the data owner's system executed by the requestor. Success in the acquisition of a report named **decrypted data** portrays the ability of every action being performed on the data to activate a condition specified in the smart contract which generates a comment to be encrypted and sent to the data owner's system by the smart contract. This property is used to effectively check for data auditability. Again, we stress on the fact that a side-block is created on the parent block that records actions pertaining to the data reported by the smart contract into the blockchain network. These are all indexed along with the data stored in the smart contract permissions database.

Algorithm 1 Smart Contract on Data

Require: *Initialization of parameters*:
getAction, getSensitivity, getRequestorID, getOwnerID, getDataID, getKey, getMetaIndex, retrieve, encrypt, comment, report, accessControl;
Ensure: *Setting up functions*:
func (getSensitivity)
func (getAction)
func (comment)
func (accessControl)
MONITORING OF PACKAGE:
for *func* (*getAction*) == *decrypt* **do**
func (*comment*) ← Potray, Data with **DataID** has been decrypted.
retrieve (*getKey*)
encrypt (*comment*)
report (*comment*||*getRequestorID*||*getOwnerID*)
end for
if *func* (*getSensitivity*) == *Low* **then**
func (*getAction*) ← Exemptions on data.
ignore
else if *func* (*getSensitivity*) == *Low* **then**
func (*getAction*) ← Not exemptions on data (violation).
func (*comment*) ← Data violation concatenated with DataID
func (*accessControl*) ← Revokes access to data.
retrieve (*getKey*)
encrypt (*comment*)
report (*comment*||*getRequestorID*||*getOwnerID*)
else {*func* (*getSensitivity*) == *High*}
func (*getAction*) ← Violation.
func (*comment*) ← Data violation concatenated with DataID
func (*accessControl*) ← Revokes access to data.
retrieve (*getKey*)
encrypt (*comment*)
report (*comment*||*getRequestorID*||*getOwnerID*)
end if

B. SMART CONTRACTS

Smart contract acts as a finite state machine executing laid down instructions when an action has been activated based on an instance. In this work, we employ smart contracts to report actions completed by a requestor on a data requested from a data owner's system. These enable data owners to completely gain assurance and control on data provenance since the entire lifeline of the sent data would be monitored in a controlled in a trustworthy environment where the data owner needs no assurance of trust from the requestor.

Data reported to the data owner's system are processed, indexed and broadcast into a blockchain network. In some instances, reports are reported and stored in a smart contract permissioned database and indexed based on the data ID of the requested and used data where sets of actions from the data owner are applicable on the data used by the requestor. We initialize sets of actions that can be applicable to any form of data retrieved from the data owner's system. The basic action sets are; read, write, delete, duplicate, move and copy. These sets of actions when performed on the data would trigger the smart contracts to send a report based on the rules established for that particular data. Monitoring of actions are conveyed in smart contract scripts by a function ***getAction***.

The sensitivity of the data is categorized into two levels which are; high and low. These levels of sensitivity are derived from processing by consensus nodes based on data sets acquired from the database infrastructure. Based on the sensitivity of a package, certain actions performed in the data are either exempted from the violations list or serve as violations. For low sensitive level for a data, the data owner, based on the requested data, can modify the smart contracts

to ignore actions to avoid reporting of extraneous data from being stored. For data with high sensitivity level, the smart contract is required to report all actions categorized in the initialization of ***getAction*** to effectively monitor operations performed on the data, ensuring the detection of breaches on data. Identities necessary to facilitate the efficient identification of unique blocks are classified for a requestor, data owner and the particular data sent. The advantage of specifying these in the smart contracts is to create an efficient means for processing and consensus nodes to match, process and verify specific blocks. Comments are generated as statements to describe the actions that were performed on the data. These are usually violation and exemption comments. A retrieve statement is paired with a ***getAction*** statement to extract an encryption key to encrypt comments that would be reported to the data owner's smart contract database. An action on ***encrypt*** is called to finalize this process. The action of sending the comments to processing and consensus nodes is instantiated by the report statement in the smart contract script. The function ***accessControl*** signifies the permissions set by the data owner that would be carried out in conjunction with the smart contract permissioned database. On violating the data contract, access to the data is revoked and pending review by the data owner who has the choice to re-grant access or retrieve data back from the requestor.

Embedding statements executable on the reception and usage of the data by requestor as an assurance of efficient operation of a smart contract is deemed important. As a matter of fact, a decryption function activated to generate a report on decryption of a package will be evident in smart contract scripts, validating the workability of the smart contract. Smart contracts on reception of violations revoke the access of the data for the requestor and sends a report which is processed accordingly and permits data owner to deal with such violations by choice through the smart contract permissions database. ***Figure 2*** is a brief description of smart contracts as a finite state machine.

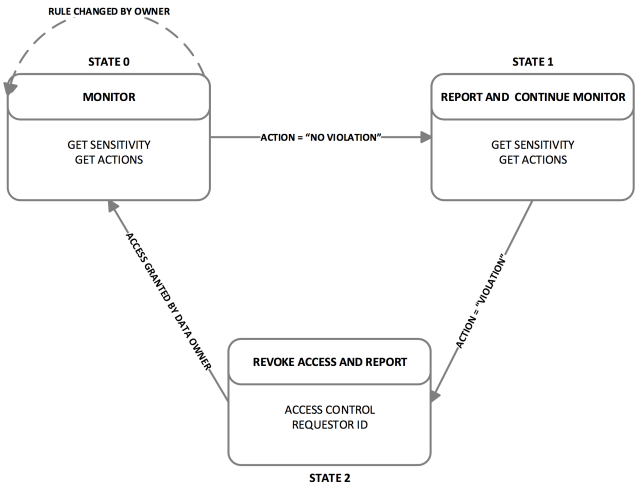

FIGURE 2. Smart Contract as a Finite State Machine (FSM).

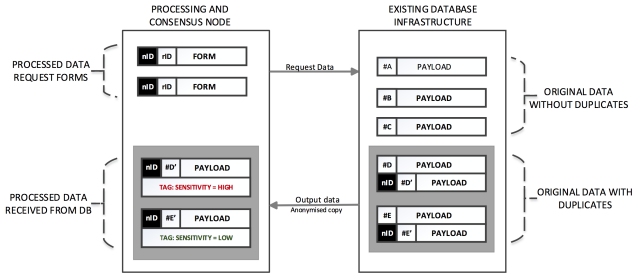

FIGURE 3. Data sharing between Existing DB Infrastructure and Processing node.

C. DATA PROCESSING AND SHARING BETWEEN EXISTING DB INFRASTRUCTURE AND DATA STRUCTURING AND PROVENANCE LAYER

Data sharing between the existing database infrastructure and the consensus node is critical for the efficient and secured functioning of the data sharing system among untrusted participating entities. Data issued by the existing database infrastructure pertaining to a request should maintain integrity and value inhibiting the ability to render the system as a contributor to already existing problems. With such needs, the data sharing methods and processing from the existing nodes and processing and consensus nodes need to be carefully designed and structured. ***Figure 3*** describes data sharing between an existing database infrastructure and a processing node.

For a verified request, the existing database infrastructure creates a duplicate of the requested data and assigns the data to the consensus node. The consensus node receives the duplicated data and creates a package out of it. The package contains a payload and a unique ID of the node responsible for processing the data along with a unique ID for the duplicated data received. The processing and consensus node responsible for processing the data verifies the received data with the form by comparing the data type with the request. The data is ranked on a scale of being highly or lowly sensitive by outlining the identifiers and quasi-identifiers. For a highly sensitive dataset, there is the need for further anonymisation. These actions performed on the data are recorded on a form transitioning into a state of being a block through processing. The result of the processed data is made available to a second processing and consensus node to validate the work done by the first processing node. Assuming validation of the processed data set is accurate, the processing node returns the data to the first node.

The processing and consensus node send a request with an attachment of data sensitivity level to the smart contract generator. The smart contract outputs a script to the node which is then attached to the current processed state of the data. Finally, the processing and consensus node outputs the result (package) of all processing to the authenticator. The authenticator encrypts the package with requestor public key and outputs timestamps to the node. The processing and consensus node records all timestamps from processing to enable optimization geared towards efficiency. In addition,

all actions performed on the data are recorded on the form including the second verifying node. The form is now processed into a block ready to be broadcast onto the blockchain network.

Algorithm 2 Packagetransaction model.

DataID: <<ID of payload, excluding signature.>>
NodeID: <<ID of processing node, including signature.>>
Data:
Hash: <<Hash of payload.>>
Sensitivity: <<Sensitivity of payload.>>
Payload
Verifier Node:
NodeID: <<Node responsible for verifying processed payload.>>
Contract and Encryption:
Smart Contract: <<Attaching smart contract to payload.>>
Authenticator Key: <<Tag key to smart contract and encrypt payload.>>
Timestamp: <<Timestamp of completed processing.>>

D. PARENT BLOCK STRUCTURE

A block is made up of a format which uniquely identifies the block. This is followed by the block size which contains the entire size of the block. The next structure is the block headers. The block header is hashed with sha256(sha256()) as done in the bitcoin headers. The block header plays a significant role in the blockchain network by ensuring immutability. By changing a block header, an attacker should be able to change all block headers starting from the genesis block in order to falsify a blocks record. This significantly ensures security on the network since there is a maximum assurance of an impossibility to achieve this task. This mechanism extensively guarantees data provenance. For malicious activities, block mismatch will alert the system of a suspicious ongoing event which triggers data forensics.

The block header contains the the data version which indicates the validation rules to follow for a particular data type. The data version gives account on the properties and the type of data being accessed. The header is also made up of the previous blocks hash which is a sha256(sha256()) hash whose function is to ensure that no previous block header can be changed without changing this blocks header. The merkle root hash forms part of the header by ensuring that non of the blocks in the blockchain network can be modified without modifying the header. This is achieved by taking the hashes of all the events in the blockchain network and appending the output to the current block. The final output is a sha256(sha256()). The header includes a timestamp of when the block was created. The header contains target difficulty which is a value of how processing is achieved by the processing and consensus nodes. This is unique to the system to make processing difficult for malicious nodes but efficient and solvable by verified consensus nodes in the system. Finally, the header consists of a nonce which is an arbitrary number the consensus nodes generates to modify the header hash in order to produce a hash below the target difficulty. The block header is therefore made up of six components.

The block has an action counter whose function is to record the total number of violation actions applicable on the accessed data in the entire block. Preceding the action counter is the transaction which is categorized into two parts that is, the timestamps and the data. The timestamps are made up of time to receive request (TTR), time taken to process request (TTP) and time to send package to requestor (TTS) whist the data part is made up of the data owner identity (OID), requestor identity (RID), sensitivity of data (Dsens), purpose of data request (DRP), processing node identity (NID) and signature of processing node (Nsig).

Finally, the structure that defines the entire block is the block locktime. This is a timestamp that records the last entry of transaction as well as the closure of a block. When conditions for this field are met, the block is ready to be broadcast into the blockchain network. The block locktime generally signifies the time the block enters the blockchain. ***Figure 4*** describes structure of the parent block.

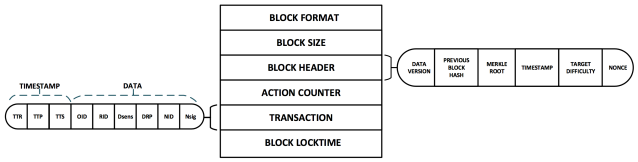

FIGURE 4. Structure of Parent Block.

E. SIDE-BLOCK STRUCTURE

A side block is made up of a format which is derived by appending a section of the main blocks ID to a generated ID by consensus node to the side block. This is followed by the block size which contains the entire size of the side block. Side blocks also have headers which are made up of six components. These are the version number which uniquely identifies the reports used to create the side block, previous side block hash, merkle root of all side blocks for a particular parent block, timestamp, target difference and a nonce. These component have same properties as their parent block but relates to side blocks.

The side block has an action counter whose function is to record the violations recorded on a single report. Preceding the action counter is the transaction which is made up of the timestamp of violation (TSV), data owner identity (OID), requestor identity (RID), violation (VLN), processing node identity (NID) and signature of processing node (Nsig). The block is timelocked and broadcast unto the blockchain by appending it to the parent block. Traces of the report and the data can now be traced. ***Figure 5*** describes structure of the side block.

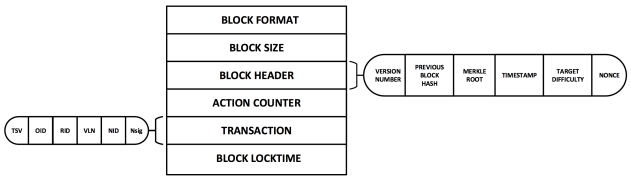

FIGURE 5. Structure of Side-Block.

TABLE 1. Latency per number of cloud service provider requests.

Number of Users	Latency(sec)
5	53.40
10	145.26
15	178.24
20	226.78
30	351.36
40	447.94
50	553.81
100	1286.73

VI. DISCUSSION

This sections evaluates the performance of implementing our system on real-case scenarios where cloud service providers share data among other cloud providers. Monitoring of action applicable on data and data revocation on instances of breaches are implemented as smart contracts in solidity. Algorithm 1 describes the formulation of a smart contract. In simulating interactions on a cloud service provider, access policies stored as records in blocks and the permissions database assigns different permissions to users for available services (access to specific data). Analysis on request of data, data retrieval and processing (includes the generating of smart contracts and tagging of smart contracts on package) and monitoring of actions on data is completed by using JMeter. Number of active requestors between 5 and 100 users for periods of 2 minutes, 5 minutes, 8 minutes, 10 minutes, 20 minutes and 30 minutes are achieved using JMeter. Vulnerabilities in MedShare are detected via access control violations on accessed data which serves as threats to the MedShare system. Vulnerability analysis is achieved if MedShare can detect any inconsistencies and violations under substantial amounts of usage load. The algorithms in MedShare should be able to detect all attacks based on the number of users thus, number of simulated attacks corresponds to the number of users simulated in MedShare data request.

Latency in MedShare has been evaluated by analysing the time taken to deliver a package after a request has been sent. This includes all steps required to process a request by all entities in the MedShare system. The latency for MedShare is represented in ***table 1*** and ***Figure 6***. An important observation on the latency is its considerable increase as request per cloud service provider increases. This is due to the trade of between achieving security over low latency. The tuple size and processing and anonymisation of data contribute to the increase in latency however, efficiency is achieved in MedShare.

Table 2 below compares our MeDShare system to other existing systems and literature presented in this paper.

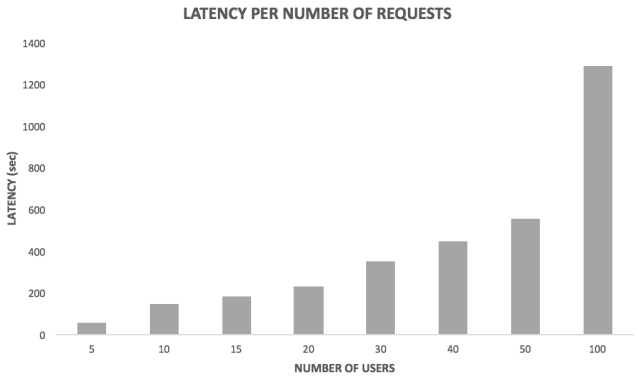

FIGURE 6. Latency per number of Cloud Service Provider requests.

TABLE 2. Comparison between proposed system and other related systems.

Metric	[26]	[27]	[28]	[29]	[30]	[31]	MeDShare
Blockchain-Based	Y	N	Y	Y	Y	N	Y
Identity Management	Y	Y	Y	Y	Y	Y	Y
Decentralised Access	N	N	N	N	N	N	Y
Centralised Access	Y	Y	Y	Y	Y	Y	Y
Distant Access	N	Y	N	N	N	Y	Y
Tamper Proof Data Audit	Y	Y	N	Y	N	N	Y
Data Access Revocation	N	N	N	N	N	N	Y

With the various metrics derived from the carefully analogy of data usage and data accumulation, a conclusion can be drawn that MedShare as compared to the systems presented has greater advantages.

VII. CONCLUSION

Data sharing and collaboration via cloud service providers is a stronghold with the increasing advancement of modern technologies driving today's society. The demand of pattern recognition and big data analysis forms a key component in this advancement as new remedies are developed from the analysis of medical data. Several methods and mechanisms have been put in place to regulate the flow of data from point(s) to point(s) as medical data in the hands of malicious entities can cause severe unthinkable damages on all parties related directly or indirectly to the data.

In this paper we design a data sharing model between cloud service providers using the blockchain. The design employs the use of smart contracts and an access control mechanisms to effectively trace the behaviour of the data as well as revoke access to violated rules and permissions on data. We analyse the performance of our system as well as compare with current cutting edge solutions to data sharing among cloud service providers. By implementing the proposed model, cloud service providers will be able to securely achieve data provenance and auditing whilst sharing medical data among other cloud service providers as well as entities such as research and medical institutions without any risk on data privacy.

REFERENCES

[1] E. R. Weitzman, L. Kaci, and K. D. Mandl, "Sharing medical data for health research: The early personal health record experience," *J. Med. Internet Res.*, vol. 12, no. 2, pp. 1–10, 2010.
[2] D. B. Taichman *et al.*, "Sharing clinical trial data: A proposal from the international committee of medical journal editors free," *PLoS Med.*, vol. 13, no. 1, pp. 505–506, Apr. 2016.
[3] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, Apr. 2014.
[4] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: Promise and potential," *Health Inf. Sci. Syst.*, vol. 2, no. 1, p. 3, 2014.
[5] H. M. Krumholz and J. Waldstreicher, "The yale open data access (YODA) project—A mechanism for data sharing," *New England J. Med.*, vol. 375, no. 5, pp. 403–405, 2016.
[6] F. F. Costa, "Big data in biomedicine," *Drug Discovery Today*, vol. 19, no. 4, pp. 433–440, 2014.
[7] J. Huang *et al.*, "A new economic model in cloud computing: Cloud service provider vs. network service provider," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.
[8] J. Huang, Q. Duan, S. Guo, Y. Yan, and S. Yu, "Converged network-cloud service composition with end-to-end performance guarantee," *IEEE Trans. Cloud Comput.*, to be published.
[9] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Comput. Netw.*, vol. 57, no. 9, pp. 2093–2115, 2013.
[10] M. R. M. Assis, L. F. Bittencourt, and R. Tolosana-Calasanz, "Cloud federation: Characterisation and conceptual model," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2014, pp. 585–590.
[11] A. O'Driscoll, J. Daugelaite, and R. D. Sleator, "'Big data', Hadoop and cloud computing in genomics," *J. Biomed. Inform.*, vol. 46, no. 5, pp. 774–781, Oct. 2013.
[12] C. L. Borgman, "The conundrum of sharing research data," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 63, no. 6, pp. 1059–1078, Apr. 2012.
[13] N. Grozev and R. Buyya, "Inter-Cloud architectures and application brokering: Taxonomy and survey," *Softw.-Pract. Exper.*, vol. 44, no. 3, pp. 369–390, Mar. 2014.
[14] M. Fazio, A. Celesti, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation: Towards interoperable storage providers," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Mar. 2015, pp. 480–486.
[15] A. M.-H. Kuo, "Opportunities and challenges of cloud computing to improve health care services," *J. Med. Internet Res.*, vol. 13, no. 3, p. e67, 2011.
[16] G. M. Weber, K. D. Mandl, and I. S. Kohane, "Finding the missing link for big biomedical data," *J. Amer. Med. Assoc.*, vol. 311, no. 24, pp. 2479–2480, 2014.
[17] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 2677–2685.
[18] D. Thilakanathan, S. Chen, S. Nepal, R. A. Calvo, D. Liu, and J. Zic, "Secure multiparty data sharing in the cloud using hardware-based TPM devices," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2014, pp. 224–231.
[19] A. N. Khan, M. L. M. Kiah, M. Ali, S. A. Madani, A. U. R. Khan, and S. Shamshirband, "BSS: Block-based sharing scheme for secure data storage services in mobile cloud environment," *J. Supercomput.*, vol. 70, no. 2, pp. 946–976, Nov. 2014.
[20] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Comput. Secur.*, vol. 42, pp. 151–164, May 2014.
[21] J.-J. Yang, J.-Q. Li, and Y. Niu, "A hybrid solution for privacy preserving medical data sharing in the cloud environment," *Future Generat. Comput. Syst.*, vols. 43–44, pp. 74–86, Feb. 2015.
[22] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proc. NIST Workshop Blockchain Healthcare*, vol. 1. 2016, pp. 1–10.
[23] R. J. Krawiec *et al.*, "Blockchain: Opportunities for health care," in *Proc. NIST Workshop Blockchain Healthcare*, Aug. 2016, pp. 1–16.
[24] M. Pilkington, "Blockchain technology: Principles and applications," in *Handbook of Research on Digital Transformations*. London, U.K.: Edward Elgar Publishing, 2015.
[25] P. T. S. Liu, "Medical record system using blockchain, big data and tokenization," in *Proc. 18th Int. Conf. Inf. Commun. Secur. (ICICS)*, vol. 9977. Singapore, Nov./Dec. 2016, pp. 254–261.
[26] T. Hardjono and N. Smith, "Cloud-based commissioning of constrained devices using permissioned blockchains," in *Proc. 2nd ACM Int. Workshop IoT Privacy, Trust, Secur. (IoTPTS)*, 2016, pp. 29–36.
[27] S. Sundareswaran, A. C. Squicciarini, and D. Lin, "Ensuring distributed accountability for data sharing in the cloud," *IEEE Trans. Depend. Sec. Comput.*, vol. 9, no. 4, pp. 556–568, Jul./Aug. 2012.
[28] G. Zyskind, O. Nathan, and A. Pentland. (2015). "Enigma: Decentralized computation platform with guaranteed privacy." [Online]. Available: https://arxiv.org/abs/1506.03471
[29] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, p. 44, 2017.
[30] S. Ferdous, A. Margheri, F. Paci, and V. Sassone, "Decentralised runtime monitoring for access control systems in cloud federations," in *Proc. IEEE Int. Conf. Distrib. Comput.*, Jun. 2017, pp. 1–11.
[31] M. M. Hassan, K. Lin, X. Yue, and J. Wan, "A multimedia healthcare data sharing approach through cloud-based body area network," *Future Gener. Comput. Syst.*, vol. 66, pp. 48–58, Jan. 2017.

QI XIA received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China (UESTC) in 2002, 2006, and 2010, respectively. She was a Visiting Scholar with the University of Pennsylvania, Philadelphia, PA, USA, from 2013 to 2014 and has published over twenty papers. She is the PI of the National Key Research and Development Program of China in Cyber Security. She is the Vice Dean of the Center for Cyber Security and currently an Associate Professor with UESTC. She won the National Scientific and Technological Progress Second Prize in 2012.

EMMANUEL BOATENG SIFAH received the B.Sc. degree in telecommunications engineering from Ghana Technology University College, Ghana, in 2014, and the M.Sc. degree in computer science and technology with the School of Computer Science and Engineering, UESTC, in 2017, where he is currently pursuing the Ph.D. degree in computer science and technology. His current research interests include blockchain technology and privacy and big data security.

KWAME OMONO ASAMOAH received the B.Sc. degree in computer science from the Kwame Nkrumah University of Science and Technology, Ghana, in 2014. He is currently pursuing the master's degree in computer science and technology with the University of Science and Technology of China. His current research includes blockchain technology and big data security.

JIANBIN GAO received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China (UESTC) in 2012. He was a Visiting Scholar with the University of Pennsylvania, Philadelphia, PA, USA, from 2009 to 2011. He is currently an Associate Professor with UESTC.

XIAOJIANG DU (SM'09) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park in 2002 and 2003, respectively. He is currently a tenured Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His research interests are wireless communications, wireless networks, security, and systems. He has over 200 journal and conference papers in these areas, as well as a book published by Springer. He has been awarded over U.S. 5 million dollars in research grants. He is a Life Member of the ACM.

MOHSEN GUIZANI (S'85–M'89–SM'99–F'09) received the B.S. (Hons.) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor and the ECE Department Chair at the University of Idaho. He has served in various positions in several academic institutions and currently serves on the editorial boards of several international technical journals. He is the Founder and the Editor-in-Chief of *Wireless Communications and Mobile Computing journal*. He has authored or coauthored over 400 publications in refereed journals and conferences. He was the Chair of the IEEE Communications Society, the Wireless Technical Committee, and the TAOS Technical Committee.