information

MDPI

Article

# BBDS: Blockchain-Based Data Sharing for Electronic Medical Records in Cloud Environments

**Qi Xia *, Emmanuel Boateng Sifah, Abla Smahi, Sandro Amofa and Xiaosong Zhang**

Center for Cyber Security, Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China; emmanuelsifah@yahoo.com (E.B.S.); ababla886@yahoo.fr (A.S.); ikwame.amofa@gmail.com (S.A.); johnsonzxs@uestc.edu.cn (X.Z.)

* Correspondence: xiaqi0769@gmail.com; Tel.: +86-28-61830061

Academic Editors: Man Ho Au, Jinguang Han, Joseph K. Liu and Ali El Kaafarani

Received: 1 March 2017; Accepted: 13 April 2017; Published: 17 April 2017

**Abstract:** Disseminating medical data beyond the protected cloud of institutions poses severe risks to patients' privacy, as breaches push them to the point where they abstain from full disclosure of their condition. This situation negatively impacts the patient, scientific research, and all stakeholders. To address this challenge, we propose 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. Our system is based on a permissioned blockchain which allows access to only invited, and hence verified users. As a result of this design, further accountability is guaranteed as all users are already known and a log of their actions is kept by the blockchain. The system permits users to request data from the shared pool after their identities and cryptographic keys are verified. The evidence from the system evaluation shows that our scheme is lightweight, scalable, and efficient.

**Keywords:** cloud computing; blockchain; data sharing; privacy; access control; electronic medical records

## 1. Introduction

The migration of medical records to cloud-based platforms has facilitated the sharing of medical data between healthcare and research institutions, enabling faster and more convenient exchange in a manner previously not possible [1]. The advantages of collaboration range from gaining new insights into already existing treatments, analysing new ones, and better management of population health. However, the risks and challenges associated with the practice are significant and cannot be ignored [2–4].

Typically, the data is stored in cloud repositories which are protected with traditional access controls in order to comply with the many policies and frameworks that govern its sharing [5–9]. These measures have several shortcomings as they have been breached time and again. First, they aim to protect the data by the penalties to be inflicted in the event of a breach; Second, users can exercise no control over the data once it leaves the sharing institution; Third, no technical barrier exists to prevent the recombination of data shared from different sources and hence the risk of re-identification of individuals remains when dealing with anonymised data sets [10]. Techniques like K-anonymity [11], $\ell$-Diversity [12], t-closeness [13,14], and others developed to guarantee privacy in such instances have not been sufficient. Patients who are unsure about adequate protection measures for the data they provide may avoid treatment altogether or make full disclosure of their medical condition a non-option.

Existing electronic medical data sharing schemes generally have one of two challenges: the first achieves good data sharing but comes with poor control. In effect, the institution that shares the data loses the ability to control it. This raises privacy concerns as there are several ways to narrow down

data sets and to re-identify individuals even where anonymizing was carried out prior to sharing. The second retains strong control of the data and remains poor at sharing. This defeats the paradigm of sharing in order to facilitate the derivation of greater value from already available data sets. Therefore, strict accountable access control to the shared cloud repository is paramount.

Generally, implementing access control in systems is usually achieved in three steps: identification, authentication, and authorization. Aside from permitting only legitimate users' entry, access control also ensures accountability: the ability to track which user carried out what action in a system. In traditional access control systems, security administrators determine which user(s) can access a particular piece of information. With that said, these current systems are more vulnerable to hacking as well as anonymous intrusions [15].

Blockchain technology is relatively new and has seen advanced developments of its most significant application in cryptocurrencies (Bitcoin currently being the best known of them all) [16,17]. Aside from the use of the blockchain technology in the aforementioned area, developers are already hard at work building applications and services (identity management, proof of existence for intellectual property, decentralised DNS services, and others) that leverage the power and versatility of the blockchain [18].

There is therefore the urgency to develop a blockchain based access control method that sufficiently controls the access to medical data stored and processed on cloud systems and securely provides efficient data sharing. Blockchain technology offers secure cryptographic techniques to identify and authenticate users and systems and thereby contrive access control in a scalable, distributed, and secure manner [19]. The blockchain in such a system will be used for data control.

In this paper, we propose a secure, scalable access control system for sensitive information. We employ secure cryptographic techniques (encryption and digital signatures) to ensure efficient access control to sensitive shared data pool(s) using a permissioned blockchain for added security and a closely monitored system. We 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. Our system succeeds where traditional access control methods of passwords, firewalls, and intrusion detection systems fail.

## 2. Related Works

In this section, research trends pertaining to access control with emphasis on the improving blockchain technology are outlined.

Guy Zyskind et al. [19] addressed privacy preservation when using third party mobile services. When signing up to a mobile application for the first time, a set of permissions (such as location, list of contacts, camera, etc.) are required to be granted for the proper functioning of the app. Unlike existing applications, the proposed platform allows only users to change the permissions in accordance with the access control policies stored on the blockchain. The proposed decentralised system consists of three entities: mobile phone users, service providers (applications), and the nodes maintaining the blockchain. In the blockchain network, two types of transactions are accepted; $T_{access}$ for access control management and $T_{data}$ for data storage and retrieval. For every user of a service, identity and the associated permissions are generated and sent to the blockchain in a $T_{access}$ transaction. Data collected from the user's mobile phone is encrypted and stored off-blockchain; only hashes of the data are stored in the public ledger. Both the user and service can query the data in a $T_{data}$ transaction.

Xiao Yue et al. [20] in their research briefly tackled access control management within the healthcare data sharing systems. The authors propose the one purpose-centric access control model. Two kinds of data users were distinguished; r-users denote users seeking to read raw data and p-users denote users who attempt to read data and retrieve results. For every data request, a transaction is made up of a requester who wants to access data of a specific category within a limited period depending on their purpose.

Guy Zyskind et al. [21] 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 the secure multiparty computation (sMPC). The different parties altogether store and run computations on data while keeping the data completely private.

Blockchain-based access control management was described in more detail in Thomas and Alex's system [22]. 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. Entities can then register—as many times as they desire—their self-asserted public key for transacting on the blockchain. Based on these public keys, miners (consensus nodes) sieve the transactions; the transactions having their origin from unknown public keys are ignored or dropped.

On the other hand, access control for Internet of Things (IoT) applications has been gaining considerable attention in recent years. Affaf Ouaddah et al. [23] proposed the FairAccess system that relies on the bitcoin blockchain technology and smart contracts for access control management in an IoT-based environment. Related to the IoT, much research on Wireless Sensor Network (WSN)-based access control has been done [24–26]. Fan Wu et al. [24] addressed the access control in secure group communication (SGC) of WSNs. In this framework, node access privileges are assigned by the network's administrator to ensure the adaptability and the flexibility of the system.

Table 1 below presents a comparison between different access control schemes presented in the literature.

**Table 1.** Comparison between proposed access control schemes from the literature.

| References | Blockchain-Based | Scalability | Identity Management | Distant-Access |
|---|---|---|---|---|
| [24] | **N** | **Y** | **N** | **N** |
| [26] | **N** | **Y** | **Y** | **N** |
| [20] | **Y** | **N** | **N** | **Y** |
| [19] | **Y** | **N** | **Y** | **N** |
| Our proposed BBDS | **Y** | **Y** | **Y** | **Y** (in case parties have already joined the system) |

Our main contribution is to provide a transparent blockchain-based Data Sharing for Electronic Medical Records system. It should be mentioned that our system relies on identity-based authentication to achieve the users' membership. Moreover, a lightweight block format has been—for the first time—proposed to improve the system's efficiency compared to the existing blockchain implementations (e.g., bitcoin).

## 3. Preliminaries

In this section, we formally define the preliminaries used in our blockchain-based scheme for electronic medical records, including the blockchain network, blocks, and cryptographic keys.

### *3.1. Blockchain Network*

The blockchain is a distributed database that contains an ordered list of records linked together through a chain, on blocks. Blocks can be described as sets of individual entities that hold information pertaining to some transactions, for example, financial transactions. The blockchain maintains a continuously growing list of records which are distributed and immutable. It is for this reason that many systems built on the blockchain technology achieve secure distribution of assets amongst untrusted clients.

For a client "A" who wants to transfer an asset to a client "B", the asset is verified by sets of nodes and ownership of such an asset is tagged to the client "A". The client transfers the asset through a channel to "B" who becomes the new owner of this asset. All these records of verification, transfer,

and change of ownership are recorded in a public database for future transactions and references. In cases where there is a malicious activity, verifying nodes can trace the attributes of the record and resolve the issues. For such systems, there is a need for a database that is highly immutable and provably secure. For transactions between clients who have no idea of the identities of each other, there is an equal need for a system to allow a trust-less but secure transaction between such parties. The blockchain is a perfect fit for resolving these problems associated with trust and change in data in cloud storage. It is highly distributed; meaning every single client part of the network has records of every valid transaction completed on the network. The properties of the blockchain that make it attractive for secure sharing of data are its immutability and "trust-less" transaction execution.

In relation to our work, the blockchain network is made up of individual blocks which are formed from events chained together from the genesis (first block) to the current or broadcast block. When blocks which contain details of an entire event (which spans from the time the user sends a request until the time he gets the data) are broadcast into the network, the blocks form a chain and can never be changed, updated, or removed. This enables data forensics and enhances data traceability in events where a user abuses the policies of data handling in the group or when there is a malicious threat in the system.

A block is made up of a single event and an event spans from the moment a request was created to when the block was broadcast into the blockchain. In instances where an authorized body needs to investigate irregularities in the system, a request is sent and access is granted to probe into the irregularities. The consensus node bears the responsibility to mine and report on the results of such irregularities. This is easy due to the blocks being linked together with an attractive property of the blockchain's immutability.

In summary, the blockchain network acts as an immutable ledger. Processed and authorised blocks are only allowed into the blockchain network if and only if the block is broadcast by an appropriate authorised entity. No other entity has the permission to broadcast blocks into the network.

### *3.2. Cryptographic Keys*

Cryptographic keys denote the sets of keys established to perform tasks pertaining to the security of a system. For the blockchain-based scheme we propose, we establish such keys which perform roles in relation to user authentication and membership, user verification, request generation, and request authentication. These keys help to guarantee a level of security of the scheme. The keys include:

- Membership issuing Keys: Generated and sent to a user who requests to join a data sharing system and allows access to the membership verification key with adequate parameters necessary to generate the transaction private and public key. Without the membership issuing key, users are not allowed to join the system.
- Membership verification Key: Used to authenticate a user's validity in a system and consents to a user's access to the membership private key. The membership verification key is used to generate the user's membership private key.
- Membership private key: Used to create a request which later develops into blocks. Without access to a membership private key, a user can never create a request.
- Transaction private key: Used to digitally sign requests created from a membership private key.
- Transaction public key: Used to verify signatures on a block. For a request whose signature cannot match to the appropriate public key, such a request is considered as invalid.

### *3.3. Membership Authentication and Verification*

For a user to join the permissioned blockchain, there is the need for membership authentication. The verification and acquisition of a user's identity used to generate specific keys in relation to the user needs to be secure through already established systems. In our work, we adopt an efficient and secure identity-based authentication and key agreement protocol which guarantees user anonymity as

a means to which membership authentication is achieved. Methods necessary to setup a new user as a member into the system is categorised into two parts.

The first section is based on an existing cryptographic techniques adopted from Wu et al.'s work [27]. The authentication phase includes system setup, key exchange, and authentication and key agreement. System setup involves generating parameters necessary to compute the shared key for the system. These parameters are drawn from the identities of the users and random numbers. Key exchange is based on sets of computations drawn from the system setup with strong emphasis on random variables that both the user and the issuer exchange with each other. Authentication and key agreement which includes the first section describes the creation of the shared, symmetric key for the user and issuer. The shared key computed by the user and issuer is used to encrypt and decrypt information pertaining to account registration and verification for the user to be a member of the system.

The second section is based on the sharing of information for account registration and verification for the user. The issuer sends an encrypted form which requires pre-set regulations (can be described as sets of policies of a system) and user details pertaining to the group concatenated with a proof of verification. Proof of verification (Pv) defines the validity of the response for the encrypted form containing user registration details. This encryption is denoted as $Enc[Fp||Pv]$ The user decrypts this encrypted file with their shared key and sends an encrypted, completed form concatenated with the valid proof of verification value, back to the issuer, $Enc[Fp||Pv]$. The issuer creates the membership issuing key from the user's identity and sends this key to the user making the user a member of the group, KMK (Key Membership issuing Key). The user requests for verification of correctness of the information provided to the issuer for the formation of the key, Correctness [KMK]. The issuer confirms the correctness of the key and sends parameters (Param) associated with the ID and a concatenation of the correctness to the user. The user uses the parameters to create a transaction key pair. The verification process described beforehand is denoted as $Verify[KMK||Param]$.

Membership verification is important to set up the rights of users to be involved in the access of data. The verification mechanisms used to achieve the blockchain-based data sharing scheme is achieved between the User and the Verifier. The user sends a request to the verifier for verification of membership into the group. The user first sends a request to the verifier for membership verification. The verifier sends a challenge with a concatenation of a random number created from the membership issuing key. The user performs computations based on the challenge and the random number and sends a response to the verifier signing the message with the membership verifying key. In addition to the response, the user sends a generated random number to authenticate the verifier's identity. The verifier compares the signature and computations with a value saved in memory associated with the shared verification keys. The verifier sends a proof of membership and a hash of the random number to the user. The user confirms the identity of the verifier and sends the verifier his transaction public key. The verifier upon receiving this stores the transaction public key in a private database. The verifier computes the membership private key from the membership verification key and sends this to the user.

### *3.4. Pool of Unprocessed Requests*

The pool of unprocessed requests is a data pool of blocks which contain requests created by the user for the purpose of contributing data or accessing data from the storage. The consensus node is the entity that fetches data from the pool for processing. Attention should be drawn to the fact that blocks found in the pool are not chained together. They are tagged with a timestamp of when they arrived in the pool. A suitable algorithm is necessary to define the order at which such blocks will be fetched and processed. A point worth noting is that the pool of unprocessed requests and the blockchain network are two separate entities which are not linked together in any way. The only characteristic between them is the consensus node.

## 4. System Design

We formulate a data sharing mechanism used by the blockchain-based medical data sharing scheme for electronic medical records. There are three entities, namely the user, system management, and storage, as shown in Figures 1 and 2. These are elaborated below.

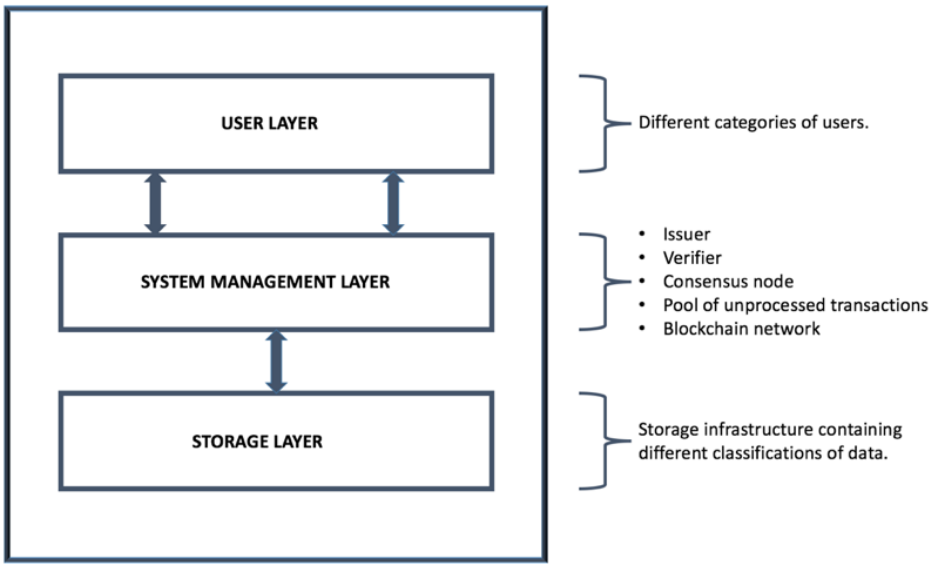


**Figure 1.** BBDS ecosystem.

- User: Users consist of individuals or organizations that want to access or contribute data from and to the closed system. The intent of most users is to help analyze data for research purposes. Examples of users can be healthcare organizations such as hospitals, research institutions, as well as universities, individual research personnel, and governmental bodies.
- System Management: The system management is composed of individual connected entities responsible for the secure establishment, efficient running, as well as optimization of the scheme. The different entities in the system management are;

    1. Issuer: The issuer forms part of the data management layer of the group by authenticating users required to join the group. The issuer sends out required details and accepts users who request to join the group based on a criterion.
    2. Verifier: The verifier forms part of the data management layer by further authenticating the user and receives the user's transaction key which is kept in a private database. The verifier later validates blocks which have been signed by the user. This enables a block's authenticity in the system to form part of the blockchain. The verifier creates and sends the user the membership private key which is used by the user to create a block.
    3. Consensus nodes: Consensus nodes fetch unprocessed blocks from the pool of unprocessed requests. The consensus node is tasked to process and verify the authenticity and details relating to a block. Processed blocks are broadcast into the blockchain by the consensus node. An important role of the consensus node is the processing and publishing of results based on irregularities in the system. The consensus node is the only entity allowed to access the pool of unprocessed requests.

- Storage: The storage layer encompasses the cloud-based data storage and processing infrastructure where data is securely kept for future reference, research, and other diverse purposes.

In this work, we refer to data requests as either

(1) Contributing data to the shared existing data stored in the cloud repository or
(2) Accessing data from the cloud repository for modification, research, or analysis purposes.

Data contributors permit the aggregation and analysis of data which leads to value derivation. Verification and auditing trails of requests are necessary to permit the continued efficient functioning of the system.

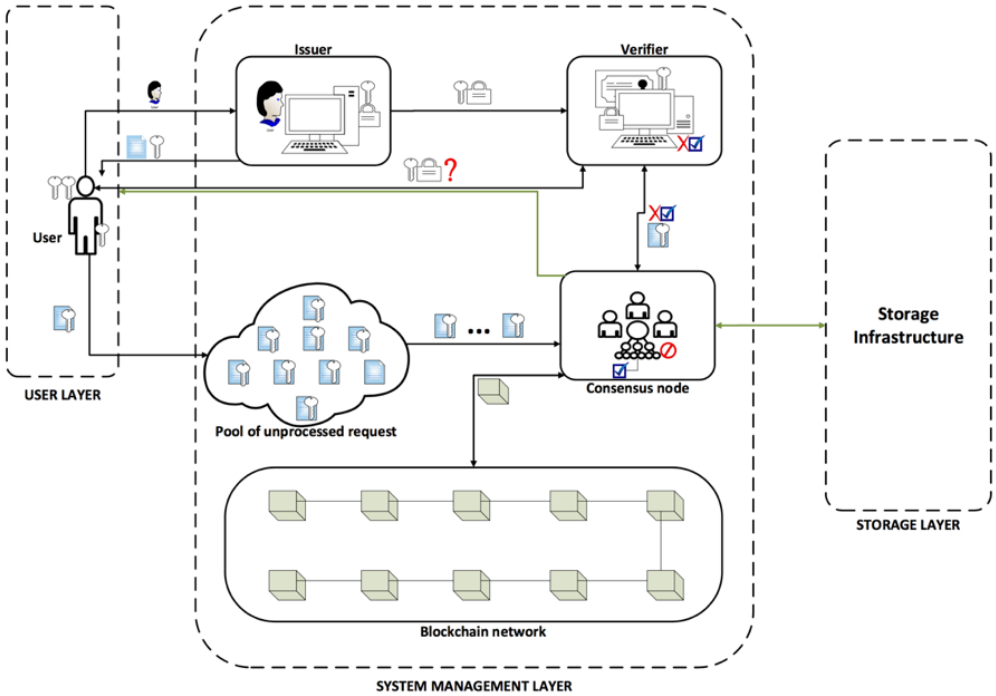


**Figure 2.** Proposed BBDS system architecture.

## 5. System Implementation

The goal of this section is to provide a framework for the BBDS construction while analysing secure structures implemented to facilitate data sharing. At the end of Section 5, Figure 3 describes the entire Blockchain-Based Data Sharing Scheme for Electronic Medical Records system logic.

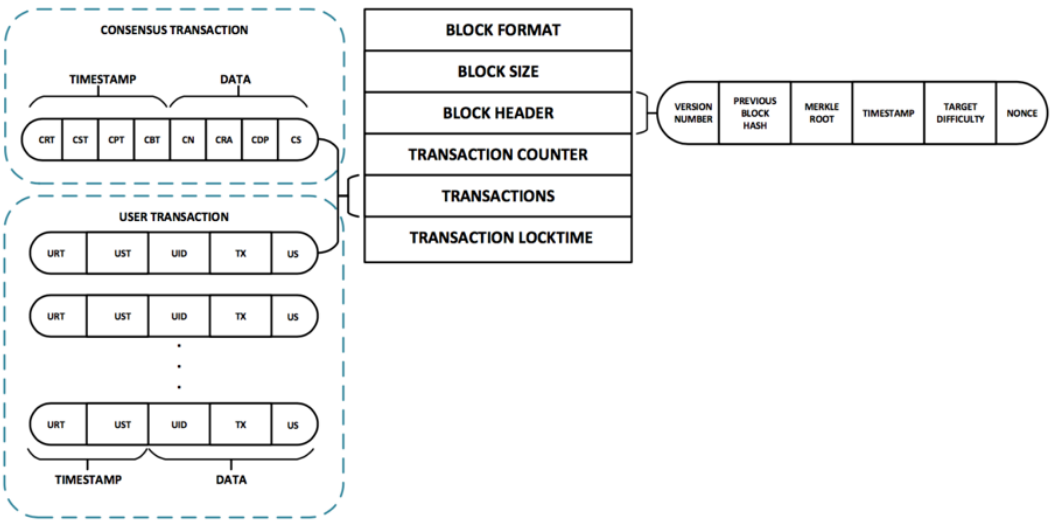


**Figure 3.** Proposed BBDS block structure.

- System setup

The issuer generates a set of membership issuing keys and membership verification keys. The issuer then shares the membership verification keys with the verifier. A user who wants to join this permissioned group sends a request to the issuer. The issuer is tasked to authenticate the user and accept the user into the group or deny user entry into the group. After successful authentication, the issuer completes this process by sending the user a membership issuing key. The user requests a key confirmation from the issuer by sending the issuer the membership key along with a generated tag of when the user entered the group. The issuer verifies whether the key is formed correctly and sends a verification key along with some other parameters to the user. The user uses the parameters to generate their transaction key pair, that is, the transaction private and public key.

The user requests for membership verification from the verifier by sending the membership verification key obtained from the issuer. The verifier verifies the user's membership in the group by sending the user a challenge. The user responds accordingly to the challenge. After successfully confirming the user's membership in the group, the verifier sends the user a membership private key. This membership private key is created from the verification keys and parameters by the verifier. The user in turn sends the verifier the transaction private key. The verifier keeps the transaction public key in a private database maintained by the verifier.

- Request file

The user sends a request for data contribution or data access into the system. The request is created by the membership private key obtained from the verifier during membership verification. Particular attention should be drawn to the fact that a block is first created when the user sends a request. The user uses the membership private key to create the block (request), sign the request with the transaction private key, and send it to a pool of unprocessed requests. The pool of unprocessed requests is made up of blocks which are not yet processed by the consensus nodes. For a request to be valid, the consensus node fetches the request from the pool and prepares it for validation.

A block is made up of a single event and an event spans from the moment a request was created to when the block was broadcast into the blockchain. In instances where an authorized body needs to investigate irregularities in the system, a request is sent and access is granted to probe into the irregularities. The consensus node bears the responsibility of mining and reporting on the results of such irregularities. This is easy due to the blocks being linked together with an attractive property of the blockchain, immutability. As shown in Figure 4, the suitable block that would suit the functions of the system is designed and highlighted.

The abbreviations used in the block structure are described in the section of Abbreviations.

A block is made up of a format which uniquely identifies the block. This is followed by the block size which contains the size of a block. The next structure is the block header. The block header is hashed with *sha*256(*sha*256()) as done in the Bitcoin headers [16]. The block header plays a significant role in the blockchain network by ensuring immutability. By changing a block header, an attacker should have to change all block headers starting from the genesis block in order to falsify a block's record. This significantly ensures security on the network since there is a maximum assurance of an impossibility of achieving this task. Block mismatch will alert the system of a suspicious ongoing event which triggers data forensics.

The block header contains the version number which indicates the validation rules to follow. The header is also made up of the previous block's hash which is a *sha*256(*sha*256()) hash whose function is to ensure that no previous block header can be changed without changing this block's header. The Merkle root hash forms part of the header by ensuring that none 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 *sha*256(*sha*256()). 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 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 generate 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 a transaction counter whose function is to record the total number of transactions in the entire block. The transaction is made up of the consensus transaction and the user transaction in relation to the purposes and processing of records as explained in the transaction section. The consensus transaction is categorized into two parts that are the timestamps and the data. The timestamps are made up if CRT, CST, CPT, and CBT whilst the data part is made up of CN, CRA, CDP, and CS. This is a standard model for a consensus transaction. The user transaction is also categorized into two parts that are the timestamps and the data. The timestamps are URT and UST whilst the data part is made up of UID, TX, and US. This forms the transaction for the user. User transactions are designed to accommodate multiple but limited events for user transaction instances that are not accounted for.

Finally, the structure that defines the entire block is the blocklocktime. 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 blocklocktime generally signifies the time the block enters the blockchain. The size of an entire block is 679 bytes. Figure 4 presents the detailed transactions' structure. Table 2 shows the size of the individual structures in the block.

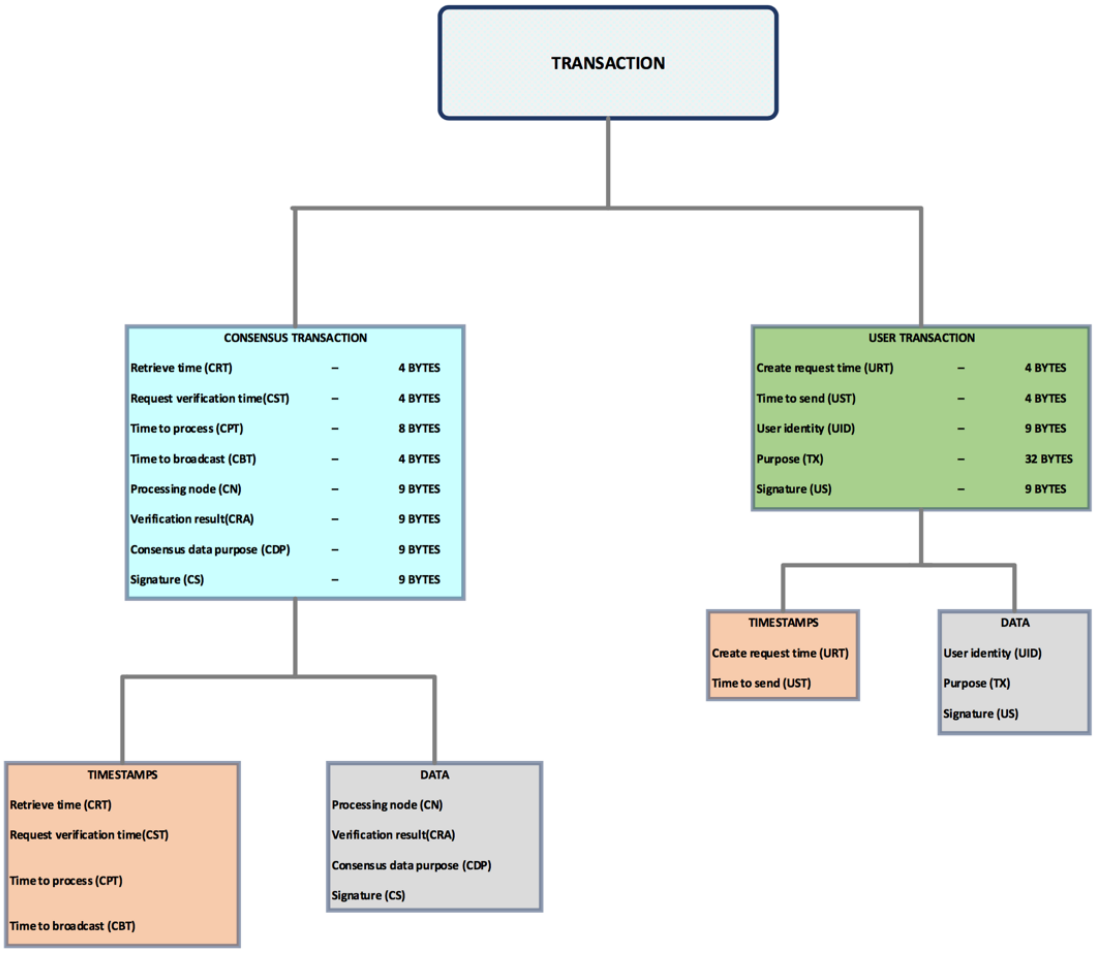


**Figure 4.** Transaction structure.

**Table 2.** Block format and component size

| Structure Name | Size in Bytes |
|---|---|
| Block format | 4 |
| Block size | 4 |
| Block header | 80 |
| Transaction counter | 9 |
| Transactions | 578 |
| Transaction LockTime | 4 |

- Grant request

The user sends a request for data contribution or data access into the system. Particular attention should be drawn to the fact that a block is first created when the user sends a request. The user uses the membership private key to create the block (request), sign the request with the transaction private key, and send it to a pool of unprocessed requests. The pool of unprocessed requests is made up of blocks which are not yet processed by the consensus nodes. For a request to be valid, the consensus node fetches the request from the pool and prepares it for validation.

A hash of when this incident occurred is recorded in the timestamp section of the block header. Again, the key triggers the block format to record a standard format in the form of a sequence of digits generated from the key to the block. The transaction layer of the block is now triggered as the request is formed as part of a user transaction, by adequately satisfying the format of the user transaction model. The URT is set along with the UID and TX to signify the purpose of the request. The user signs the data, which is recorded in US and sends the request. A timestamp recording the exact time the data was sent is captured by UST. These actions complete a single transaction for a user.

The consensus node fetches a request that triggers the target difficulty, nonce, and consensus transaction. The consensus node solves a mathematical puzzle whose answer is updated by the target difficulty and nonce. This allows the consensus node to process the request. The completion of this triggers the consensus transaction since the consensus node sends the request for user verification by the verifier. CRT records the exact time of the data retrieval by the node. The action of sending the signature for verification is recorded at CST and the action for verification of a user is captured by CRA. The total time for processing is recorded by CPT with the processing node's identity executed by CN.

- Access file

After a user's request (data contribution or data access) is granted, access is given to the user pertaining to the data request or contribution and the verified request is ready for broadcast which is recorded by CBT with a record of purpose and the user in addition to the purpose for the creation of this block. This is recorded by CDP. A signature to verify proof of processing is recorded by CS. During the processing, the format layer of the block is completed by appending a sequence number to the generated format by the user key to identify the current position of this block. As part of processing, the version number of the block is recorded in the header. The hash of the previous block is also recorded in the header and the Merkle root is formed which includes the current block. The hash of the header is formed, readying the block for broadcast. The block length is recorded and the BlockLockTime then locks the block with a timestamp. At this time, there is an assurance of the block being a constituent of the blockchain network and forming part of a chain.

The block is now part of a chain system and adds to the height of the blocks in the blockchain system. For occurrences where other blocks are formed after this block, the hash of the header of the last block in the network is taken and appended in the header of the newer block, which is yet to be part of the chained blocks in the network. The whole process is shown in Figure 5 that bellows.

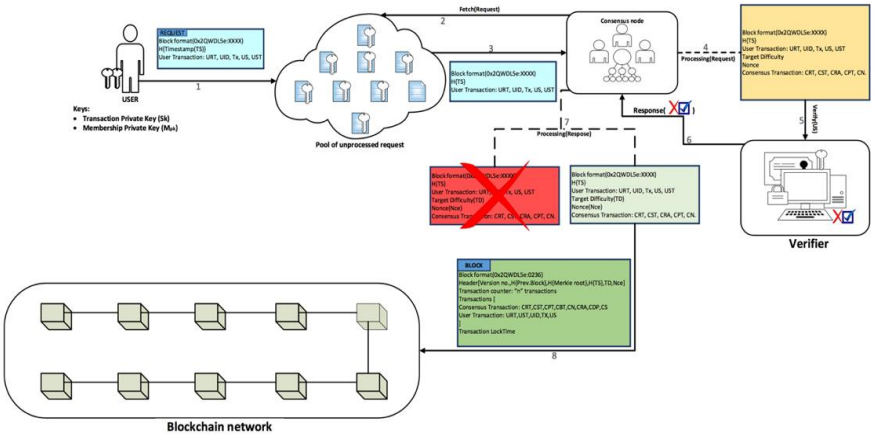


**Figure 5.** Proposed BBDS system logic.

## 6. Protocols Used to Complete the System

As demonstrated in Figure 6, the interactions between both user-issuer and issuer-verifier are described in this section.

### *6.1. User-Issuer Protocol*

For communication between the user and the issuer, there is the need for authentication and verification. This section describes the protocols necessary to setup a new user as a member into the system. Part of the formation of the protocol is adopted from Wu et al. [27] regarding their research based on an efficient and secure identity-based authentication and key agreement protocol which guarantees user anonymity.

***Part 1:***

The protocol between the user and the issuer is divided into two parts. The first section is based on existing cryptographic techniques adopted from Wu et al.'s work [27]. The authentication phase of our protocol from the adopted work is further divided into three phases:

A. System setup phase (generating parameters necessary to compute the shared key for the system)
B. Key exchange phase (sets of computations and keys that both the user and the issuer exchange with each other)
C. Authentication and key agreement phase (creating the shared key for the user and the issuer)

***Part 2:***

The shared key computed by the User and Issuer is used to encrypt and decrypt information pertaining to account registration and verification for the user to be a member of the system:

1. The issuer sends an encrypted form which requires pre-set regulations and user details pertaining to the group concatenated with a proof of verification. Proof of verification defines the validity of the response for the encrypted form containing user registration details, $Enc[Fp||Pv]$.
2. The user decrypts this encrypted file with their shared key and sends an encrypted, completed form concatenated with the valid proof of verification value, back to the issuer, $Enc[Fp||Pv]$.
3. The issuer creates the membership issuing key from the user's identity and sends this key to the user making the user a member of the group, KMK.
4. The user requests for verification of correctness of the information provided to the issuer for the formation of the key, Correctness [KMK].
5. The issuer confirms the correctness of the key and sends parameters associated with the ID and a concatenation of the correctness to the user.
6. The user uses the parameters to create a transaction key pair, $Verify[KMK||Param]$.

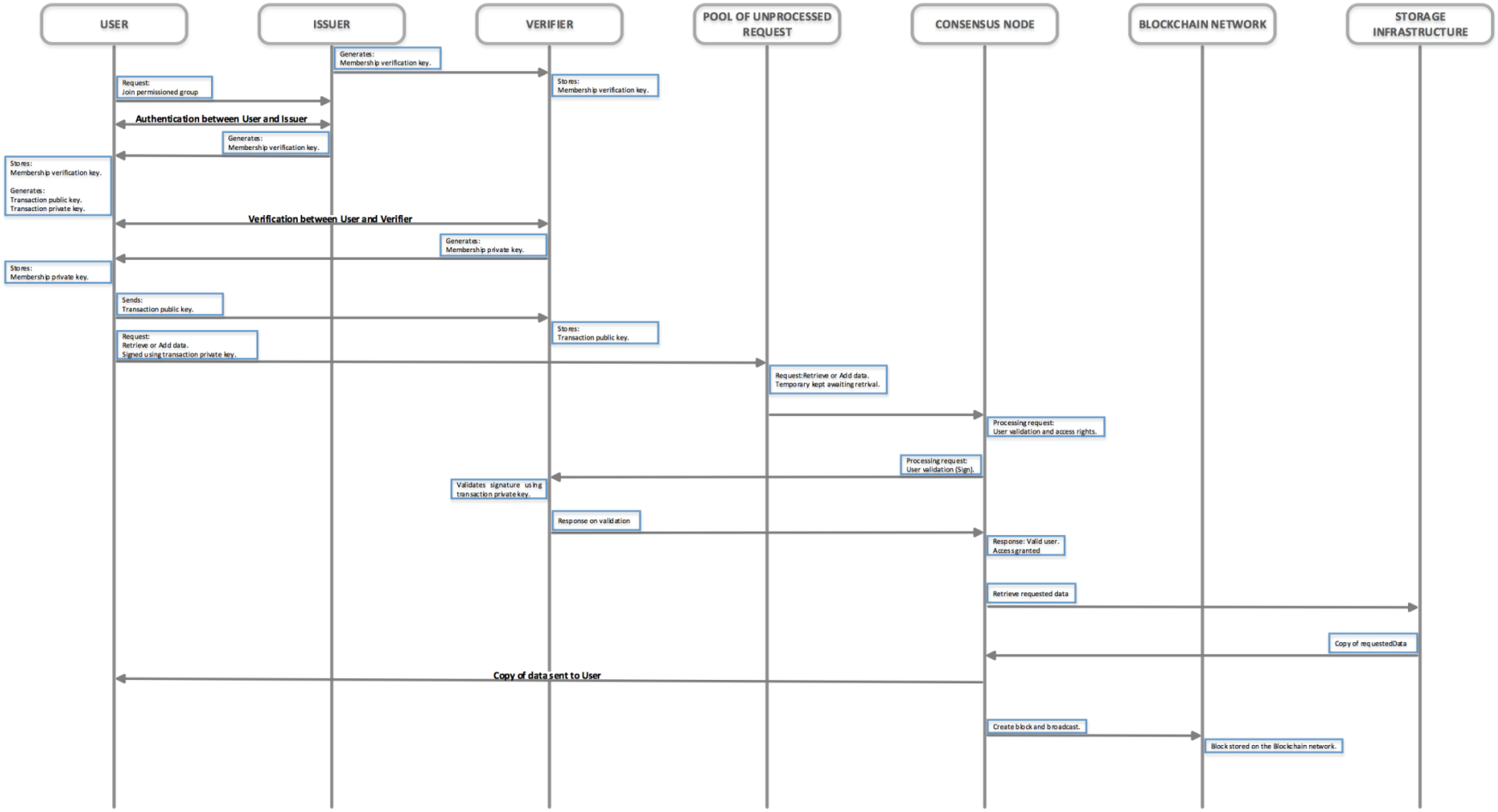


**Figure 6.** Protocols used to complete the system.

### *6.2. User-Verifier Protocol*

The user sends a request to the verifier for verification of membership into the group. This section discusses the protocols necessary for the establishment of verification by the user and verifier:

1. The user first sends a request to the verifier for membership verification.
2. The verifier sends a challenge with a concatenation of a random number created from the membership issuing key.
3. The user performs computations based on the challenge and the random number and sends a response to the verifier signing the message with the membership verifying key. In addition to the response, the user sends a generated random number to authenticate the verifier's identity.
4. The verifier compares the signature and computations with a value saved in memory associated with the shared verification keys.
5. The verifier sends a proof of membership and a hash of the random number to the user.
6. The user confirms the identity of the verifier and sends the verifier his transaction public key.
7. The verifier upon receiving this stores the transaction public key in a private database. The verifier computes the membership private key from the membership verification key and sends this to the user.

## 7. Evaluation

Scalability is one of the major challenges facing the current bitcoin networks whose underlying technology is the blockchain. The block structure described in Section 5 is proposed as a solution to the scalability problem.

To determine the size and scalability of the blockchain network (compared to the blockchain currently implemented within bitcoin) in this system, tests based on calculated assumptions derived from different vendor transactions per unit time have been simulated. For clarity, the transactions of existing vendor networks have been explained and the number of users on this system has been adopted to exaggerate the entire size of the blockchain in our blockchain-based data-sharing scheme for a given period of time.

The fast-growing Bitcoin network, processes on average a single transaction per second with a theoretical maximum of seven transactions per second [28]. Considering the number of transactions processed by other vendors, implementing such numbers on the Bitcoin blockchain network leads to scalability issues. Bitcoin's blockchain has a size of over GB, with an increase of several GB in 2015. Considering an increased throughput of about 2000 transactions per second, the supplementary transactions would result in a growth of about several PB/year. At 150,000 transactions per second, the size of the blockchain would overwhelm the network. The generation of such huge amounts of data per year is due to the fact that a block size is 1 MB in Bitcoin's blockchain.

The number of transactions processed by Visa is 2000 transactions per second with a maximum of 10,000 transactions per second at peak times such as holidays. With respect to other vendors, the number of transactions processed by Twitter is 5000 transactions per second with a maximum of 15,000 transactions per second at peak times. Advertising networks, trading networks, and email networks process a minimum of 500,000 to 2,100,000 transactions per second [29]. Ideally, a carefully designed block structure on a blockchain network will support these massive amounts of data generated each second without being concerned with the scalability of the networks.

By carefully structuring the block size in our system, we adopt the use cases of user transactions per second to create an over-exaggerated value for users sending such requests per second. The importance is to prove the scalability of the blockchain network in our system in exaggerated conditions. The following formulas are used in the calculation for all vendor transactions per second and can be used for real case scenarios.

- Calculations

**Tx(B×ts)** used to calculate data size per second
**Tx(B×60(tm))** used to calculate data size per minute.
**Tx(B×36×102(th))** used to calculate data size per hour.
**Tx(B×864×102(td))** used to calculate data size per day.
**Tx** defines the number of transactions sent per period of time
**B** defines the block size, 679 MB
**ts** defines the time in seconds
**tm** defines the time in minutes
**th** defines the time in hours
**td** defines the time in days

- Example

Assuming the system consists of 1000 users with a maximum of 200 users sending transactions each second, the size of the blockchain network would be: 135 bytes/s, 7.77 MB/min, 466.23 MB/h, and 10.29 GB/day.

- 200(679×1) = 135,800 bytes per second = 132.62 KB per second
- 200(679×60(1)) = 8,148,000 bytes per second = 7.77 MB per minute
- 200(679×36×102(1)) = 488,880,000 bytes per second = 466.23 MB per hour
- 200(679×864×102(1)) = 11,733,120,000 bytes per second = 10.29 GB per day

Using the example above as a guide, Table 3 shows an assumed over-exaggerated number of user transactions per second. The total size in bytes of the results shown in the table below are outputs pertaining to blocks generated in this system. Data sizes generated are represented in megabytes (MB), gigabytes (GB), terabytes (TB), and petabytes (PB). The transaction column illustrates the number of transactions per period of time. Data generated per period in relation to the design block represents the amount in size for the blockchain network over a period of time.

**Table 3.** Estimated blockchain network growth.

| | **Data Generated Per Period in Relation to Designed Block** | | | |
|---|---|---|---|---|
| **Transaction** | **Per Second** | **Per Day** | **Per Year** | **Per 10 Years** |
| 2000 | 1.29 MB | 108.84 GB | 38.79 TB | 387.90 TB |
| 10,000 | 6.47 MB | 545.91 GB | 194.59 TB | 1.95 PB |
| 15,000 | 9.71 MB | 819.28 GB | 292.03 TB | 2.92 PB |
| 500,000 | 323.77 MB | 26.68 TB | 9.74 PB | 97.40 PB |
| 2,100,000 | 1.33 GB | 112.22 TB | 40.96 PB | 409.60 PB |

## 8. Conclusions

In this paper, we proposed a blockchain-based mechanism to mediate access between users and a pool of shared (sensitive) data. Compared to the Bitcoin blockchain network, we constructed a scalable (redesigned to allow speedy transactions) and lightweight blockchain to demonstrate the efficiency of our design which permits data sharing in a secure manner and protects the privacy of data. In the proposed system, communication and authentication protocols and algorithms between entities were not fully investigated. It would be interesting to extend this work by fully exploring these in future studies. We state that the architecture described in this paper is a top layer of the blockchain-based access control system that is under implementation and testing. In our future work, an experimental study will be conducted to improve the system's efficiency and to obtain empirical data for further studies.

**Acknowledgments:** The Fundamental Research Funds for the Central Universities (ZYGX2015J154), Key research and development projects of high and new technology development and industrialization of Sichuan Province (2017GZ0007). The paper was approved by Ke Huang after the authors collectively reviewed it.

**Author Contributions:** Qi Xia and Emmanuel Sifah Boateng conceived the idea while Abla Smahi designed the block structure and logic flows. Xiaosong Zhang contributed to the feasibility and analysis of the ideas expressed. Sandro Amofa wrote the paper.

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

## Abbreviations

| | |
|---|---|
| CRT | Consensus retrieve time |
| CRA | Consensus verification result |
| CPT | Consensus time to process |
| CST | Consensus request verification time |
| CBT | Consensus time to broadcast |
| CDP | Consensus data purpose |
| CN | Consensus Processing Node |
| CS | Consensus signature |
| URT | User retrieve time |
| UST | User time to send |
| UID | User identity |
| TX | User purpose |
| US | User signature |

## References

1. Longo, D.L.; Drazen, J.M. Data Sharing. *N. Engl. J. Med.* **2016**, *374*, 276–277. [CrossRef] [PubMed]
2. Davis, J. 7 Largest Data Breaches of 2015. *Healthcare IT News*. 11 December 2015. Available online: www.healthcareitnews.com/news/7-largest-data-breaches-2015 (accessed on 14 April 2017).
3. Higgins, K.J. Healthcare Data Breaches From Cyberattacks, Criminals Eclipse Employee Error For The First Time. *Information Week DarkReading*. 5 July 2015. Available online: http://www.darkreading.com/attacks-breaches/healthcare-data-breaches-from-cyberattacks-criminals-eclipse-employee-error-for-the-first-time/d/d-id/1320315 (accessed on 14 April 2017).
4. IBM-Security. *Reviewing a Year of Serious Data Breaches, Major Attacks and New Vulnerabilities: Analysis of Cyber Attack and Incident Data from IBM's Worldwide Security Services Operations*; IBM Security: Somers, NY, USA, 2016.
5. Sladić, G.; Milosavljević, B.; Konjović, Z. Modeling context for access control systems. In Proceedings of the 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (SISY 2012), Subotica, Serbia, 20–22 September 2012; pp. 37–42.
6. Elliott, A.; Knight, S. Start Here: Engineering Scalable Access Control Systems. In Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies, Shanghai, China, 5–8 June 2016; pp. 113–124.
7. Maw, H.; Xiao, H.; Christianson, B.; Malcolm, J. A Survey of Access Control Models in Wireless Sensor Networks. *J. Sens. Actuator Netw.* **2014**, *3*, 150–180. [CrossRef]
8. Brucker, A.D.; Hang, I.; Lückemeyer, G.; Ruparel, R. SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes. In Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT'12), Newark, NJ, USA, 20–22 June 2012; pp. 123–126.
9. Chen, H.; Bhargava, B.; Zhongchuan, F. Multilabels-based scalable access control for big data applications. *IEEE Cloud Comput.* **2014**, *1*, 65–71. [CrossRef]
10. Anonymous. Data Breaches Cost the Healthcare Industry an Estimated $6.5 Billion. *Micrographics* **2011**, *29*, 3–5.
11. Sweeney, L. K-Anonymity: A model for protecting privacy. *Int. J. Uncertain.* **2002**, *10*, 557–570. [CrossRef]
12. Machanavajjhala, A.; Gehrke, J.; Kifer, D.; Venkitasubramaniam, M. L-Diversity: Privacy beyond k-anonymity. In Proceedings of the International Conference on Data Engineering, Atlanta, GA, USA, 3–7 April 2006; Volume 2006, p. 24.

13. Ninghui, L.; Tiancheng, L.; Venkatasubramanian, S. T-Closeness: Privacy beyond k-anonymity and L-diversity. In Proceedings of the International Conference on Data Engineering, Istanbul, Turkey, 11–15 April 2007; pp. 106–115.
14. Soria-Comas, J.; Domingo-Ferrert, J. Differential privacy via t-closeness in data publishing. In Proceedings of the 2013 11th Annual Conference on Privacy, Security and Trust (PST 2013), Tarragona, Spain, 10–12 July 2013; pp. 27–35.
15. Ausanka-Crues, R. Methods for Access Control: Advances and Limitations. Available online: https://www.cs.hmc.edu/~mike/public_html/courses/security/s06/projects/ryan.pdf (accessed on 14 April 2017).
16. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: www.bitcoin.org (accessed on 14 April 2017).
17. Sasson, E.B.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. Zerocash: Decentralized anonymous payments from bitcoin. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–21 May 2014; pp. 459–474.
18. Schneider, J. Blockchain—Putting Theory into Practice. Available online: https://t.co/CLJJf0tGp0 (accessed on 14 April 2017).
19. Zyskind, G.; Nathan, O.; Pentland, A.S. Decentralizing privacy: Using blockchain to protect personal data. In Proceedings of the 2015 IEEE Security and Privacy Workshops (SPW 2015), San Jose, CA, USA, 21–22 May 2015; pp. 180–184.
20. Yue, X.; Wang, H.; Jin, D.; Li, M.; Jiang, W. Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control. *J. Med. Syst.* **2016**, *40*, 218. [PubMed]
21. Zyskind, G.; Nathan, O.; Pentland, A. Enigma: Decentralized Computation Platform with Guaranteed Privacy. *arXiv* **2015**.
22. Hardjono, T.; Pentland, A.S. Verifiable Anonymous Identities and Access Control in Permissioned Blockchains. Available online: www.w3.org/2016/04/blockchain-workshop/interest/hardjono-pentland.html (accessed on 14 April 2017).
23. Ouaddah, A.; Elkalam, A.A.; Ouahman, A.A. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. *Adv. Intell. Syst. Comput.* **2017**, *520*, 523–533.
24. Wu, F.; Pai, H.T.; Zhu, X.; Hsueh, P.Y.; Hu, Y.H. An adaptable and scalable group access control scheme for managing wireless sensor networks. *Telemat. Inform.* **2013**, *30*, 144–157. [CrossRef]
25. Wang, Y.W.Y.; Attebury, G.; Ramamurthy, B. A survey of security issues in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **2006**, *8*, 1–23. [CrossRef]
26. Huang, H.F. A novel access control protocol for secure sensor networks. *Comput. Stand. Interfaces* **2009**, *31*, 272–276. [CrossRef]
27. Wu, L.; Zhang, Y.; Xie, Y.; Alelaiw, A.; Shen, J. An Efficient and Secure Identity-Based Authentication and Key Agreement Protocol with User Anonymity for Mobile Devices. *Wirel. Pers. Commun.* **2016**. [CrossRef]
28. Luu, L.; Narayanan, V.; Baweja, K.; Zheng, C.; Gilbert, S.; Saxena, P. SCP: A Computationally-Scalable Byzantine Consensus Protocol For Blockchains. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 1168.
29. McConaghy, T.; Marques, R.; Muller, A.; de Jonghe, D.; McConaghy, T.; McMullen, G.; Henderson, R.; Bellemare, S.; Granzotto, A. BigchainDB: A Scalable Blockchain Database (DRAFT). Available online: https://pdfs.semanticscholar.org/1c0c/5640e2efcd32480f94020bf857c261acdae4.pdf (accessed on 14 April 2017).