






# Interoperability and Synchronization Management of Blockchain-Based Decentralized e-Health Systems

Sujit Biswas , *Member, IEEE*, Kashif Sharif , *Member, IEEE*, Fan Li , *Member, IEEE*, Zohaib Latif, Salil S. Kanhere , *Senior Member, IEEE*, and Saraju P. Mohanty , *Senior Member, IEEE*

**Abstract**—E-health systems have witnessed widespread usage in the last few years, mainly due to advancements in health monitoring hardware and the availability of remote diagnostic services. Given the numerous benefits, there are two main challenges: management of the e-health ecosystem and security and privacy of sensitive data. The former results from nonunified, redundant, and often replicated information across numerous independent e-health service providers. While the latter stems from sensitive information stored in centralized systems that can be compromised. Blockchain has emerged as a promising technology, which can be used to secure access and privacy of data and provide an umbrella management solution to large-scale distributed and decentralized enterprise systems. In this article, we present a unified system for migrating independent conventional e-health systems to a single blockchain-based ecosystem. More specifically, we address the issues of difference in data structures for conventional relational databases and blockchain file databases. The solution describes the conversion process and synchronization of information in a unified system for large-scale e-health data. The implementation and analysis show that significant improvements in data storage, access control, and seamless migration can be achieved.

**Index Terms**—Blockchain (BC), distributed ledger technology, e-health ecosystem, interoperability, Internet of Things (IoT), synchronization.

## I. INTRODUCTION

**E**-HEALTH systems (EHS) are one of the best applications of smart devices and networks, such as the Internet of Things (IoT), providing important life-altering services to people. The impact of these services is reaching people not only in developed urban areas but also in rural and especially in

Manuscript received June 16, 2019; revised March 7, 2020; accepted April 15, 2020. The work of Fan Li was supported in part by the National Natural Science Foundation of China under Grant 61772077 and Grant 61432015 and in part by the Beijing Natural Science Foundation under Grant 4192051. Review of this manuscript was arranged by Department Editor K.-K. R. Choo. (*Corresponding authors: Kashif Sharif; Fan Li.*)

Sujit Biswas, Kashif Sharif, Fan Li, and Zohaib Latif are with the School of Computer Science and the Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, Beijing Institute of Technology, Beijing 100081, China (e-mail: sujitedu@bit.edu.cn; kashif@bit.edu.cn; fli@bit.edu.cn; z.latif@bit.edu.cn).

Salil S. Kanhere is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: Salil.Kanhere@unsw.edu.au).

Saraju P. Mohanty is with the Department of Computer Science and Engineering, University of North Texas, Denton, TX 76203 USA (e-mail: Saraju.Mohanty@unt.edu).

Color versions of one or more of the figures in this article are available online at <https://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TEM.2020.2989779

underdeveloped countries. The statistics show that the global e-health market has grown ten times from 2013 to 2018, i.e., patients from approximately 0.35 to 7 million, related devices and services value approximately 440.6 million to 4.5 billion [1], and has a compound annual growth rate of 30.8% [2]. It is estimated that 75.44 billion IoT devices will be utilized globally by 2025 [3]. The medical providers increasingly employ remote communications and monitoring technology to reduce costs and improve the quality of care. According to the World Health Organization, 87% of countries already have taken initiatives for e-health and it is expected that forthcoming 5G technology will increase it further. However, the trained manpower (physicians) is not increasing at such a rate [4], which highlights the need for automated, efficient, and unified EHS.

In an EHS, the Internet of Medical Things (IoMT) devices and networks are major contributors due to their rapid adoption and deployment. They are mostly used for remote monitoring of patients by healthcare providers. Body area and body sensor networks are an integral part of IoMT and comprise heterogeneous devices that generate multidimensional data. In addition to IoMT devices, on average a person may use 2–3 IoT devices, for example, the smartphone with numerous sensors, a smartwatch, and perhaps a smart exercise wearable. Hence, every EHS user (patient) utilizes a large number of IoMT devices, which may be developed by various manufacturers, and may not comply with any specific data storage standard. Hence, the data generated are extremely heterogeneous and mostly stored in relational databases (RDB) supporting structured query language (SQL) at a centralized server. Moreover, in real-world scenarios, there are numerous e-health service providers working independently, as a result, a patient's common history is recorded on several different servers, which violates atomicity of data [5]. In order to build a national level (or similar) health service to provide better medical facilities, it is imperative that the information systems of these health service providers are either merged or interlinked. This will not only remove the data redundancy but also keep atomicity, which ultimately improves data collection and analysis for disease prediction and prevention at a larger scale. It will directly benefit the patients, as their medical history will be readily available at all service locations, regardless of their ownership.

### A. Challenges of EHS

Merging EHSs not only makes a unified national EHS but also can create an autonomous national health data center, which is already a norm in different advanced countries [6], [7]. By

adopting the Industry 4.0 electronic health record [8], [9], an autonomous unified EHS can increase the speed of big data processing. A unified EHS introduces challenges of 5V [10], i.e., volume, velocity, variety, variability, and value. In a unified EHS, the data are generated by ubiquitous devices in a variety of different formats, and for different purposes. They can be as simple as a prescription slip, and as complex as MRI images with associated diagnostic reports. Similarly, the size of these data may vary from a few kilobytes to tens of megabytes per element. To ensure real-time digest and analysis of these by service providers requires a high number of transactions per second (i.e., velocity). Moreover, patients' medical records are not only private but also valuable, which demands that appropriate access control policies should be in place. From this discussion, we conclude two basic needs for modern EHSs: First, individual EHS must be unified (or seamlessly interlinked) to provide a national-level (and beyond) efficient healthcare facility. Unification and interlinking in the presence of different types of data, central servers, and RDBs are a highly complex task. Second, utilization of modern security and distributed technologies, to ensure that either individual EHS or unified EHS do not allow illegal access to data or leakage of information while ensuring efficiency, availability, and scalability.

### B. Blockchain (BC) Technology

BC [11], [12] in recent times has emerged as a potential solution that can integrate the conventional EHS to improve access control, traceability, unification of information, availability, and efficiency [13]. BC is a distributed ledger technology where *NoSQL*-based file database ledger is held by every participating peer, which can accept diverse unstructured data. Peers are interconnected to each other in a peer-to-peer transport layer security (TLS) supported network. From transaction initialization to finalization, every step is validated in a decentralized manner. The transaction approval process depends on consensus mechanism (i.e., maximum member peers vote), whereas votes are cast based on the contract between transaction issuer and receiver. A transaction is stored in a block along with other transactions with proper encryption. Finally, newly created blocks are linked with the latest block, making a chain of sequential immutable blocks (i.e., the BC). The chain, which is stored at every peer, is synchronized for every new block. Due to immutability, changes to existing information are impossible, whereas the distributed and redundant nature of ledger avoids a single point of failure.

### C. BC-Based EHSs

Several research efforts have been done to integrate e-health with BC from different aspects. We discuss these in the next section, however, to the best of our knowledge, no work directly addresses the migration from conventional and individual EHSs to BC-based unified EHS. Even if a unified system is not created, individual EHS cannot directly shift to a BC-based system. Some of the major challenges in this regard are as follows.

- 1) Existing centralized EHS store data in RDB, whereas BC uses a file database. Moreover, the DB schema of different EHS does not have a direct one-to-one mapping.

- 2) Due to restriction on transaction size in a block of BC, it is impossible to store complete medical imagery as part of the chain.
- 3) Due to real-time transactions at a mass scale, it is challenging to migrate all medical history of all patients to BC ledger.
- 4) In an EHS, it is quite possible that some medical documents are paper based, e.g. electrocardiogram output.

Hence, the only way to digitize them is to store as images, which is a nonreal time process. Based on these issues, proper migration and synchronization of SQL supported RDB to NoSQL-based file database is a challenging and complex task. Perhaps an ideal solution is to utilize the strengths of BC for immutability, unification, and access control while using RDB for indexing and off-chain storage of large medical records, such as imagery.

### D. Contributions

This article precisely addresses the challenges mentioned earlier of integrating centralized EHS with the BC-based EHS network. More specifically, we present a complete solution for migration and synchronization from traditional to BC-based database systems. It is capable of supporting heterogeneous storage data formats, services, and applications, for efficient sharing of e-health records having flexible data structures with semantic metadata. The synchronization is performed in real-time with strict access control for all participating entities. The major contributions of this article are as follows.

- 1) A BC-based novel framework for unified EHSs with compatibility to work with conventional centralized e-health platforms.
- 2) Analysis of challenges and issues faced while merging SQL-based RDB and NoSQL-based file databases of BC ledgers.
- 3) A secure and efficient mechanism for synchronization of conventional EMR with BC ledger.
- 4) Comprehensive emulation based performance analysis of a working hybrid model.

It is important to highlight that this article is not focusing on the representation structure of EMR, rather it addresses the storage structure of data. At the very basic level, every EMR element is stored to optimize the storage capacity and efficient retrieval, which is then converted to standard representational formats for healthcare professionals and exchange at the application level. This application-level processing is not disturbed (but rather made more efficient) by the proposed work.

The rest of this article is organized as follows. Section II discusses exiting works in the field of EHSs using BC. Section III elaborates on the proposed framework and its system design. Section IV gives the details on implementation and processes. Section V presents the testbed observation and performance discussion. Finally, Section VI concludes this article.

## II. RELATED WORKS

Several research efforts have been done to integrate BC into EHSs. [14] elaborates different problems and benefits of BC in EHS. The authors focus on scalability issues generated by medical data; however, their solution does not consider the

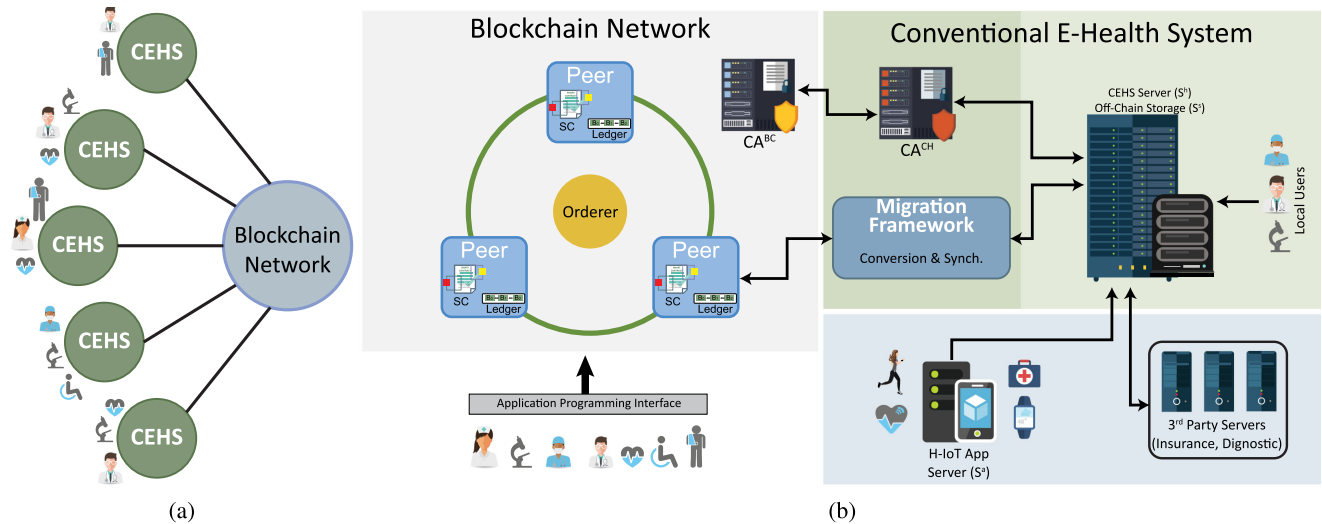


Fig. 1. Overall framework of a unified BC-based EHS. (a) Unified BC Network. (b) System Model.

heterogeneity of data or its migration. Liang *et al.* [15] focused on mobile healthcare user's privacy and data storage security challenges. They proposed a tree-based integrity management method for facilitating scalability and efficient data processing. However, access control is a complex process and the solution does not address all possible concerns, such as limiting the privileges of past physicians, medical staff, or diagnostic centers to newer data. Considering the privacy issues, a single user that acts as both the data owner and the data retriever, and uses authentication key agreement schemes for accessing or submitting transactions has been proposed in [16]. However, the patient's data should be accessible by several physicians and healthcare professionals. Consequently, to protect data leakage, a multiuser supported searchable e-healthcare system has been proposed in [17]–[19]. However, it is not a BC solution but does present an interest key generation and exchange mechanism, which can be utilized in BC-based EHS. Similarly, Yang *et al.* [20] used a multiuser access key to provided access control in a cloud-based conventional EHS. Liu *et al.* [21] proposed a BC-based framework for removing intermediaries to ensure secure and efficient EHS transactions. Although data security and users privacy issues are partially considered; however security threats from both internal and external users and record storage have not been clarified. In a scenario where a patient leaves the hospital (either after recovery or transfer to another hospital), they may not wish their new medical data to be accessible by past physicians. Yang and Ma [22] considered this issue and proposed to re-encrypt the patient's data with a new key. However, this may be a costly solution as all past data have to be re-encrypted every time, following which the new keys made available to future physicians.

Most of the related works consider security and privacy issues of EHS, but very few present a comprehensive migration solution. More specifically, to the best of our knowledge, no work addresses the synchronization of RDB in conventional EHS with the file-based system of BC technology. A research project Catena [23] has done some credible work to support distributed immutable relational databases for BC. Although

not directly focusing on EHSs, but its technical features of SQL supported immutable distributed ledger may be utilized in EHS. Similarly, translation of SQL queries to NoSQL-based BC ledger has not been addressed in the literature. Some available tools (i.e., Apache Sqoop [24] and DataX [25]) and [26] show generic conversion mechanisms. However, their integration with a BC-based EHS is an open challenge.

The motivation of this article is to lay a foundation framework for individual conventional EHS migration toward a BC-based unified healthcare system. We address the challenge of exchanging data between RDB and BC, which will be a major issue in modern BC-based EHS. We show a novel method for storing heavy medical images in an off-chain storage, which is linked to individual transactions stored in blocks.

### III. SYSTEM DESIGN

The basic architecture of the proposed solution is shown in Fig. 1(a). Each conventional e-health system (CEHS) with a stand-alone EMR management system has to be connected to a common BC network. An important aspect here to consider is that the migration of individual CEHS to individual BC systems is possible; however, the challenge in connecting multiple BC systems together is far greater than connecting multiple CEHS to each other. Hence, in this article, we propose the use of a unified national-level healthcare network, which is primarily a BC network. We assume that such a BC network will be maintained by the health regulatory authorities in collaboration with private healthcare service providers, making it a private/permission business BC network. This assumption is quite realistic, as many countries across the globe have invested in better health facilities at the national level, and can maintain peers and communication networks. The integrated BC-based EHS can be divided into three subsystems, as shown in Fig. 1(b). It shows an individual CEHS and its interaction with the business BC network. Besides, it also shows other third-party systems, such as diagnostic labs, insurance providers, IoMT data collection servers, etc., which regularly interact with the service providers. All these systems



provide data that ultimately becomes part of electronic medical records. In the sections below, we individually explain the purpose and working of each subsystem, whereas the detailed explanation of the migration framework is given in Section IV.

### A. BC System

The BC system comprises minimum three peers, which maintain the distributed immutable shared ledger. The network represents a private business BC architecture, which will interlink each existing CEHS. In addition, it also allows access to other remote users, which may not be complete CEHSs, but should be able to access data or create transactions, as shown in Fig. 1. The core elements of the BC network are described in detail below.

1) *Certificate Authority (CA<sup>BC</sup>)*: All elements interacting with the BC network must register with the CA server, which is the sole authority to generate different certificates and signatures. These elements include users, orderer, devices, channels, and peers. Any user  $u_i \in U$  can be categorized into two subgroups based on their role in the EHS. In this article, we classify them as: patients  $Pt_i \in U$  and service providers (SP). The latter are usually physicians  $Ph_i \in SP$  or service devices  $SD_i \in SP$ , where  $i = \{1, 2, 3, \dots, n\}$  and  $SP \subset U$ . It is important to note that there can be other types of users, however, as they are not in the scope of this article, hence we have not defined them. CA<sup>BC</sup> generated certificates and credentials are used to ensure proof-of-identity for secure transactions. The user registration process starts by sending a request to CA<sup>BC</sup> using application interfaces (APIs) and completes by the approval of the peer administrator. Before merging the CEHS and BC network, all existing users must register with CA<sup>BC</sup>.

2) *Peers (P)*: The BC network comprises at least three peers to ensure consensus for all transactions. An authorized  $u_i$  can forward its transaction ( $T_i$ ) request to  $P_i \in P$ , which then becomes a leader in the consensus process and requires some peers to endorse the transaction. Here,  $P' \subseteq P$  are the endorsing peers. Each  $P_i$  holds the smart contracts as well as a copy of the common ledger. The consensus process is based upon a smart contract between relevant parties, e.g.,  $Pt_i \leftrightarrow Ph_j$ . In this article, the consensus protocol does not affect the working of the proposed solution. The unified BC ensures that all peers are using the same algorithm, hence any optimized algorithm [27] in this regard can be deployed.

3) *Smart Contract (SC)*: It is a self-executing programming script, which outlines a predefined agreement between trading parties, and is installed on the peer. In the context of an EHS, it defines the access control and privileges of different users for obtaining patient records or updating them. Practically, it is impossible to have a unique contract for each patient and physician, hence we suggest the use of generic contract templates, which can be changed on a need basis. For instance, a smart contract can be considered for  $Pt \leftrightarrow Ph$ ,  $Ph \leftrightarrow SD$ , or between a patient and diagnostic center, etc. As this article is patient-centric, hence  $Pt_i$  controls permission privilege to limit the access rights for other parties (Ph, SD, etc.).

A user's application executes a smart contract through three functions. `Init()`, to start and initialize an SC or chaincode on a channel; `Invoke()`, for sending a  $T_i$  proposal to a  $P_i$ ; and `Query()`, which executes the ledgers current state without

writing anything to the ledger. It is important to note that SC is not always between a single patient and a single physician. Different scenarios may require the same contractual conditions for a group of users. Hence, a smart contract  $SC_i^{Pt \leftrightarrow Ph}$  can be  $\forall Ph$  who are treating  $Pt_i$  currently. However, individual permission for different  $Ph_i$  can be defined through function parameter of a contract (see Section IV-B). We primarily considered three types of coarse accessibility.

- 1) *One-to-one* access: The agreement between two individuals is mandatory, such as  $Ph_j$  and  $Pt_i$ . It is important to note that, it is not necessary to change the SC every time a patient changes a physician, rather the access list of physicians in SC can be updated from the world state.
- 2) *One-to-many* access: Simultaneous multiple service providers may require access to a single patient record. We consider this very similar to the one-to-one access, as the solution is patient-centric. The patient can allow individual  $Ph_j$  associated with different CEHS, who can then use the unified BC to query the respective EMRs.
- 3) *Many-to-many* access: For certain research and analysis purposes, anonymized medical data of multiple patients can be accessed by multiple authorized research centers. The anonymization of information is enforced through a smart contract; however, prior consent of patients can be done through individual smart contracts. In this article, we have not addressed this issue as it is purely based on smart contract.

4) *Orderer*: In a BC network, orderer works as a transaction collection entity, mainly responsible for block creation and adoption of transactions. For example, a set of transactions  $T$  is approved by  $P' \subseteq P$  through consensus, and is collected for by orderer to be committed in a block  $B_i$ . Orderer ensures that all  $P_i \in P'$  have provided valid signatures, and then broadcast  $B_i$  to all peers. Following this,  $B_i$  is linked with  $B_{i-1}$  of chain and the ledger transaction process is completed.

5) *Ledger (L)*: The ledger comprises a *chain of blocks*, which are immutable, sequenced records of transactions. It maintains a tamper-resistant record of all state transitions in the network, which are the result of chaincode invocations (i.e., transactions) submitted by participating parties. Ledger data are separated into two parts as *world state* and *BC*. The world state maintains the current key values of a set of ledger states and the last transactional record summary for any user. The BC itself is the collection of blocks with transactions.

6) *Channel*: Channel is used as a secured communication path for users to communicate with peers and orderer, or for intracomponent communication in a BC network. In this research, we propose a user-centric channel management system, hence a channel  $C_i$  is created during the registration of user  $u_i$ , whereas other members of  $C_i$  are subsequently added subject to the approval of  $Pt_i$ . Channel configurations are modified with changes in access policy and recorded in genesis block, which is the first block of the chain (see Section IV-A.1).

### B. Conventional EHS

Most of the existing EHSs maintain and provide medical services through a centralized server  $S^h$ . Data are collected from all different user groups, such as Pt, Ph, SD, etc., at this

single point, as shown in Fig. 1. Here, we discuss the core components and their possible interaction with other subsystems in the architecture.

1) *Users/Devices*: Any kind of existing participants in the CEHS are considered as local users. They will become a user of BC through the registration process with  $CA^{BC}$ . Following the same grouping, as described earlier, the patients, physicians, healthcare providers, and devices are appropriately given validation credentials. It is important to note that any user of CEHS, which is not able to successfully register with  $CA^{BC}$ , remains local user and cannot execute chain code.

2) *Application Server ( $S^a$ )*: Wearable IoMT devices are deployed on the patient's body and usually controlled through smartphones acting as gateways. As most of the IoMT devices do not have enough computation and storage resources, all collected data are forwarded and processed in a centralized application server. The important point to note here is that the application server  $S^a$  is not usually the central server of CEHS. Several third-party IoMT devices are available, which push data to the third-party  $S^a$ . Subsequently,  $S^h$  can receive data from  $S^a$  if compatible APIs are available.

3) *CEHS Server ( $S^h$ )*: It is the main server of a CEHS, which is centralized and holds SQL supported relational database. At the time of bridging  $S^h$  with a BC network,  $S^h$  only acts as an off-chain storage server  $S^s$  for medical images or heavy data, which cannot be stored in BC ledger, and for indexing the stored data.

4) *Access Control Firewall*: A logical entity that controls direct CEHS server access of locally connected users or service providers (i.e., Ph, Pt, SD, etc.). For all  $u_i$ , the authentication is approved through the BC network with the collaboration of  $CA^{BC}$  and  $CA^{CH}$ .

### C. Migration Framework

The migration framework, as shown in Fig. 1(b), is not a subsystem as it resides inside the CEHS; however, it is the major focus of this article. Hence, it is important to introduce it as part of the system design and elaborate its internal components, whereas Section IV explains its working in detail.

Existing CEHS servers serve a dual role. They act as the application connecting point, as well as the database for all medical information. This database is usually an SQL supported RDB, whereas the BC network supports a file-based NoSQL database. Since CEHSs are merging with BC network, hence the challenge is to exchange information between systems seamlessly. A smooth migration platform is extremely important for bridging these different information stacks. Furthermore, as a BC always works with real-time data, hence it is not possible to simply migrate the existing RDB to a ledger. The objective of migration framework is to create a hybrid database, by synchronizing the data coming to/from both sources. In a unified EHS, all data retrieval or appending is done as a transaction, where large datasets are stored in the conventional server as RDB and smaller information (which can be stored as strings) is made part of the transaction stored on the ledger. The migration framework is responsible for parsing the query and transaction, decide the execution areas, and finally publish the combined result. There are two main elements in the migration panel: The certificate

authority of CEHS ( $CA^{CH}$ ) and digital data converter (DDC) for synchronization, overview of which is given below, whereas the working details and internal architecture are explained in Sections IV and IV-C specifically.

1) *Existing Certificate Authority ( $CA^{CH}$ )*: Every EHS has an individual certificate authority that shares certificates and other credentials with BC network  $CA^{BC}$ . Existing users of CEHS ( $v_i \in V$ ) who have signed up with  $S^h$  are considered as authenticated  $CA^{CH}$  members. At the initialization time, a respective admin may take initiatives for this one-time migration task for registering  $V$  as  $U$  on  $CA^{BC}$ . BC network considers  $CA^{CH}$  of any CEHS as a trusted authority. Ideally  $\forall(v_i \in V)$  under  $CA^{CH}$  should migrate, hence  $V^{CA^{CH}} \subset U^{CA^{BC}}$ . However, it is possible due to authentication (or other) reasons,  $\exists(v_j \in V)$  are not registered with  $CA^{BC}$ . In order to keep the system realistic, the framework allows such users but limits them to that specific CEHS. They may be able to access the local RDB, but cross-platform access to other CEHSs through BC will be prohibited.

2) *Digital Data Converter*: This is a logical entity working in cooperation with  $S^h$ , and executes any kind of query from/to NoSQL supported BC network. The main objective of DDC is to forward respective BC queries to  $S^h$  in an executable format. It is also able to parse JavaScript Object Notation (JSON) text to SQL query for retrieving the archived data (see Section IV). Unlike RDB, every NoSQL data type is different in structure. However, in either case, there is always a common identifier that is used by DDC as key.

## IV. SYSTEM WORKFLOW

The proposed framework presents complete access control, transaction processing, and interoperability of centralized CEHSs with BC network. The BC network is primarily responsible for access control; more precisely, monitoring and control of who is accessing whose data and how much of it. This access control is defined through a special kind of transaction. The system defines two types of specialized transactions: First, transaction for access policy management ( $T_P$ ), used for recognizing who is accessing whom, and second, transaction for data ( $T_D$ ), used for sending/retrieving the actual payload. As some  $T_D$  may be incompatible with BC data structures, this challenge is resolved by storing partial transaction in off-chain storage. Interoperability of off-chain (i.e.,  $S^s$ ) and BC is solved through the bridging framework of DDC.

### A. Access Control

BC is being widely used to enforce security and privacy for big data networks, such as IoT, ML, etc. It has also been used for access control besides ledger management [28]. Although from an EHS perspective, access control is quite different from other use cases, as patients access rights change frequently. However, in BC, member's access rights are verified and controlled through smart contracts, which does not support frequent changes. In this article, we propose a channel-based patient-centric access control policy in addition to smart contracts, which allows a dynamic list of users (e.g., physicians) to access the data. By specifying a channel per patient, access is largely controlled with a *compound key*, created using channel-members' keys,

**Algorithm 1:** Compound Key Generation Process.

---

**Input:**  $(C_i^{id}, Pt_i^{id})$   
**Output:**  $(\text{Compound.key}(\text{sck}))$

- 1: Initialize  $SP^{Pt_i}[] \setminus \text{list}$
- 2:  $sk_{\varepsilon_{nc}}^{Pt_i}, pk^{Pt_i} \leftarrow \text{cryptogen}(\varepsilon_{nc}())$
- 3:  $pk^{Pt_i} \rightarrow \forall_{SP_i \in SP^{Pt_i}[]}$
- 4:  $\text{Resp}[], pk^{SP_i}[] \leftarrow \text{Response}(SP_i^{Pt_i})$
- 5: **if** all  $\text{Resp}[]$  are valid **do**
- 6: |  $\text{sck} \leftarrow \text{Compound.key}(pk^{Pt_i}, pk^{SP_x}[], sk_{\varepsilon_{nc}}^{Pt_i})$
- 7: **end**
- 8: **Return**  $\text{sck}$

---

in order to provide the highest level of security to that channel. The process of creating channels and compound key is explained below.

1) *Channel Creation:* A channel  $C_i \in C$  is created per patient through a policy transaction, which is stored as *config* block within the genesis block. The transaction contains core information about the channel such as: *version*, a number which changes with every modification, and *permission*, policies that govern whether or not access/modification of elements are permitted. After the creation of a channel, the following operations can be performed on it.

- 1) *Update:* If for any patient  $Pt_i \exists (C_j \in C)$ , then a new *version* is added to  $C_j$  using the same process as given earlier. The condition of  $\exists! C_j : Pt_i$  must hold.
- 2) *Deletion:* Usually a channel will never be deleted. A channel can only be deleted if the patient is being removed from BC network.

2) *Compound Key Generation:* The channel is owned by the patient while other service providers are members. With the changes in members, policies are changed, and the channel is recognized to have a new version. Since channel *id* is unchanged, to restrict unauthorized past members from accessing the channel, a compound key is generated for every new version. Algorithm 1 shows the secret compound key (*sck*) generation process. A list of service providers is generated through policy transaction, which are selected by the patient (see Section IV-B). In line 2,  $CA^{BC}$  generates key pairs by  $\text{cryptogen}()$  (which is a common tool used in CA). Lines 3 and 4 send the public key of  $Pt_i$  to every selected service provider, which means that the patient is authorizing them to access data. In response, service providers return their public key and a response value, which is recorded as a list. Finally, if all responses are accepted, then a compound key is generated, which is a combination of public keys of all participants and secret key of  $Pt_i$ . It is then shared with all channel members. It is important to note that  $\text{compound}()$  generates an irreversible hash value as the compound key.

### B. Transaction

In a patient-centric EHS, all users initiate transactions through their perspective APIs, which in turn are responsible to communicate with the peers. A  $Pt_i$  initiates a transaction through its channel, whereas the channel members require prior permission to use that channel to execute any transaction related to  $Pt_i$ . To

ensure secure and restricted access to the data,  $Pt_i$  approves all members as well as individual permission set for each member. Application originates two kinds of transactions, i.e., policy transactions ( $T_P$ ), which define the permission privileges, and data transaction ( $T_D$ ) for any registered user-generated medical data. Details of both types are given below.

1) *Policy Transaction ( $T_P$ ):* During the signup process, each user's identity related credentials (i.e.,  $u_i^{id}$ ,  $u_i^{keys}$ ,  $u_i^{certs}$ ) are generated by  $CA^{BC}$ . It is the responsibility and right of  $Pt_i$  to grant appropriate permission for access to different data elements (i.e., blood test report, MRI image, physician report, medication slip, etc.) associated with it. Let  $Sr_x^{SP_i}$  be a list of such data elements, where  $x$  is the total number of such elements, allowed to be accessed by a specific service provider  $SP_i$ . All these permissions are granted through a policy transaction generated by  $Pt_i$ . Similarly, any changes to already instantiated access policy or channel members are also done through a new  $T_P$ . Let  $T_P^{Pt_i}$  be a new policy transaction generated by  $Pt_i$  for its channel  $C_j^{Pt_i}$ . Due to the fact, that a policy change for the channel has occurred, it is mandatory that a new secret compound key (*sck*) is generated using Algorithm 1. Following this, the new key is shared with current/updated members of  $C_j^{Pt_i}$ , and stored in the *config* transaction of the genesis block. This update also triggers the *version update* of channel. This process gives two direct benefits. First, this access to the channel is restricted for all service providers, which may have an old *sck*. Second, the version identifier is always deployed and stored in every transaction, hence it allows historical data to still accessible by an approved  $SP_i$  of that time.

2) *Data Transaction ( $T_D$ ):* In the proposed unified EHS, a data transaction  $T_D$  may be of three types: Transactions that have simple string based data that can be stored as part of a transaction in ledgers, transactions that have heavy medical documents that cannot be part of the ledger, and query transactions that allow retrieval of stored medical data from the ledger or off-chain storage. Each one is described in detail as follows.

- 1) *Transaction without heavy data:* A simple data transaction  $T_D^i$  can be invoked by any  $u_i \in U$  in a channel where they have sufficient privileges to do so. For example,  $T_D^{Ph_i}$  can be a simple physician's prescription, or  $T_D^{SD_i}$  can be heart rate monitored by a sensing device. As the data are relatively small and textual in nature, hence it can be part of a transaction. Users execute the transactions by their application through a customized channel that is generated for that specific group of users as discussed earlier. More specifically, the application executes  $\text{invoke}()$  function with all parameters, as given in Listing 1. Every channel member accesses the channel using the latest *sck* corresponding to the version of the channel. If the channel approves access for  $T_D^i$ , it is forwarded to endorsing peers  $P'$ , where  $P' \subseteq P$ . The payload, which is the transaction proposal, comprises user credentials and chaincode information. Additionally, the actual data payload is part of  $\text{arg}[]$ , as shown in line 6. After peer verification of credentials, including chaincode, signatures of  $P'$  are returned to the invoking application. Finally,  $T_D^i$  is forwarded to the orderer for commit to block. Orderer collects other transactions from different applications for that particular time slot, cross verifies all credentials (i.e.,



signatures of endorsers and CA certificates), and valid transactions are added into the block and sent to all peers. Peers again verify the signatures of all approving peers and orderer, and formally commit the block to ledger. Acknowledgment of the transaction is then passed to the application.

- 2) *Transaction with heavy data* ( $\check{T}_D$ ): Any kind of  $T_D$  with images or documents that needs to be stored is considered as heavy data transaction and symbolized as  $\check{T}_D$ . BC solutions do not support images or data files as part of a regular transaction. For example, the maximum transaction size of Bitcoin is 1 MB [29], which is very less to store an image. However, medical images are regular transactional data in an EHS based on BC. In order to address this issue, in this article, we propose the use of off-chain data storage [30]. Client application fundamentally generates  $\check{T}_D$  along with the medical images. However, once it determines that the data are heavy, it generates a hash code of the images and creates a file pointer to the original image, which will be stored in off-chain. The hash code ensures that the image whenever retrieved can be checked for immutability, whereas the file pointer stores the information of where in the off-chain it is stored. This can be a physical path to the encrypted directories or other storage locations. The application separates original images from  $\check{T}_D$ , and only keeps the hash and pointer address of the data as part of  $T_D$  in BC. BC ledger stores the hash of the images and the pointer that links BC ledger to the off-chain storage [31]. We also propose that the application strip-off any identifiable data from the images as a precaution. Hence, the BC network forms a shell around the off-chain storage, where the encrypted and anonymized images have no meaning without proper transaction information securely present in the ledger.

- 3) *Query* ( $\check{T}_D$ ): In a BC-based system, any query to retrieve patient information is executed through a transaction on the BC. Any registered user can initiate such a transaction  $\check{T}_D$  shown in Listing 2. It is forwarded to the application associated peer through the authorized channel. During the invoke process, the channel ensures the user's authorization and access right for the specific information, whereas the respective peer ensures the contract between parties. If every verification is positive, only then the query transaction is approved. Finally, the requesting user ( $u_i$  or  $SP_j$ ) receives result of  $\check{T}_D$ .

An important point to note is that if the response of  $\check{T}_D$  contains hash and pointer to heavy data, the application executes another nested query to retrieve images from off-chain storage. In order to retrieve such data, there is no overhead of the peer or the BC network. The server  $S^h$  executes this peer-approved  $T_D^{qry}$  nested query in RDB ( $S^s$ ) using the given file pointer. Details of this are given in the following section.

### C. Interoperability

To enable real-time synchronization, smooth and seamless exchange of transactions is crucial between BC and CEHS

```

1 invoke() {
2   let tx_id = this.connection.newTx_id();
3   let TxData = {
4     chaincodeId: 'PatientVsServiceProviders',
5     fcn: 'upload_msg()',
6     args: [],
7     txId: tx_id,
8     chainId: 'chainid',
9   };
10  return this.connection.submitTx(TxData);
11 }

```

Listing 1. Transaction payload function.

```

1 query() {
2   let QryData = {
3     chaincodeId: 'PatientVsServiceProviders',
4     fcn: 'Search',
5     args: ['']
6   };
7   return this.connection.query(QryData);
8 }

```

Listing 2. Transaction query function.

networks. Conversion from and RDB structure to a ledger-acceptable transaction dynamically is extremely complex. In this article, we use the DDC module to enable such synchronization. Fig. 2 shows the overall communication flow as well as a conversion process. The user-generated transactions are sent to peer through an application, where it verifies whether the transaction has associated heavy data or not. If it has heavy data, it creates a hash and generates a storage reference path and invokes it to peer, otherwise, it directly invokes the transaction. Peer verifies the transaction against respective SC and collects the consensus approval from other peers. After consensus formation, peer approves the transaction, sends an acknowledgment to the application and adds it to the ledger. Consequently, the application sends heavy data to the off-chain storage ( $S^s$ ) for storage with the help of the migration system (DDC). CEHS server maintains an index of images and key to the BC transaction.

Unlike an RDB structure, the BC data structure is quite different as it has to maintain extra metadata information. It is very important to understand that the data structure of BC and RDB cannot be directly stored into each other, and proper conversion is required in a unified format to achieve convergence. In the following sections, we first describe the structure of BC transaction and block, followed by a unified conversion principle in the form of *triples*. Finally, we show the conversion process with an example.

1) *Data Structure*: In the complete system, there are two unique data structures involved: Blocks of the chain and tables of an RDB. Here, we elaborate on the features of both for clarity of understanding.

a) *Blockchain*: It can simply be viewed as a linked list where a block  $B_i$  is linked with  $B_{i-1}$ , through a hash value. Every block has an individual identity as well as a hash of all transactions in that block. Fig. 3 shows the core structure and elements of a transaction inside a block, as explained in the following.

- 1) *Block header*: Comprises block sequence in integer nonce ( $N^{B_i}$ ), Hash( $T$ ) where  $T$  are all existing transactions in the block, and immediate previous block Hash( $N^{B_{i-1}}$ ).

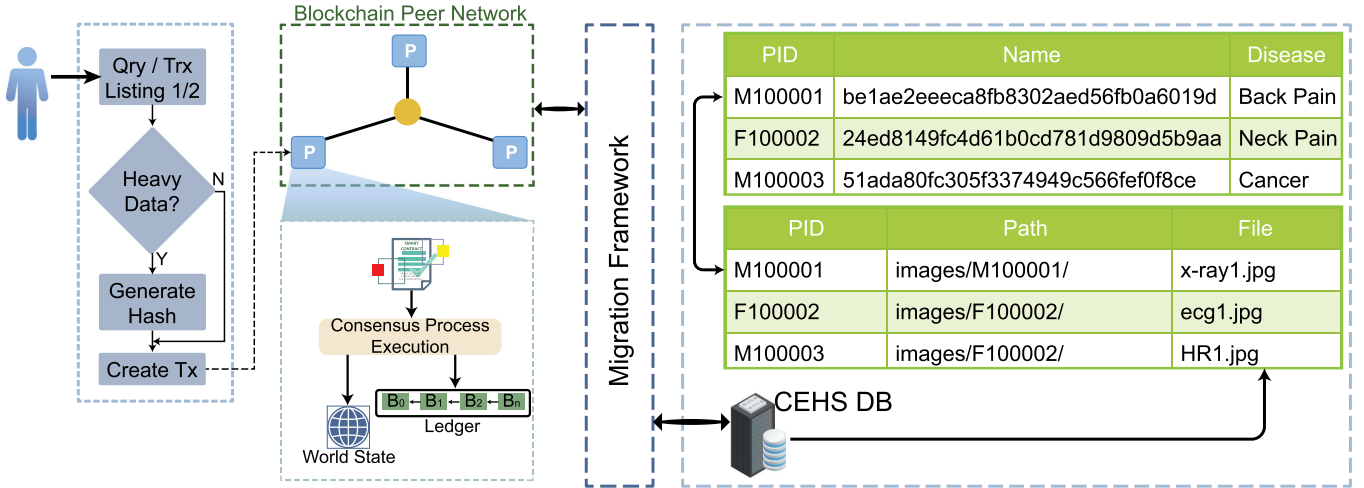


Fig. 2. Workflow of transactions.

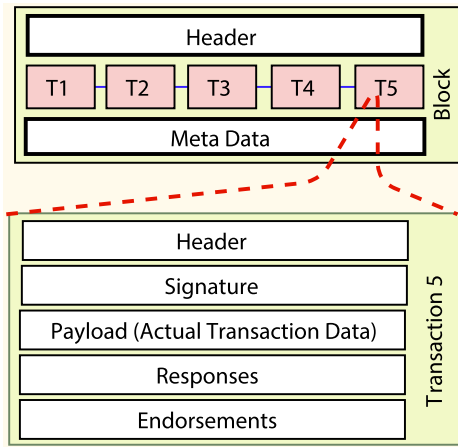


Fig. 3. Block and transaction structure in a BC.

- 2) *Transactions/block data*: Contains a list of all transactions (i.e.,  $T_1, T_2, \dots$ ), where each contains the following.
  - *Header* contains essential metadata about the transaction, such as chaincode name, version, etc.
  - *Signature* is a cryptographic signature of the client generated by their private key.
  - *Payload* is the actual transactional data (transaction proposal) supplied by clients application.
  - *Response* is the result of chaincode execution reflected as a Read–Write set, and also indicates a valid/invalid transaction.
  - *Endorsement* is the collection of signatures of endorsing peers.
- 3) *Metadata*: It contains all information regarding a block such as endorsers and orderer consent, signature, keys, and other transactional credentials.

b) *Relational database*: The records in an RDB are stored in a tabular structure, where a unique (compound) key identifies a particular record and links it to other relations. In contrast to BC properties, RDB is usually based on atomicity, consistency,

isolation, and durability properties [32]. The records are usually inserted, extracted, or updated using SQL, whereas in BC, the ledger is based on NoSQL, which is more suitable for big data operations [26].

2) *Transformation Principle*: Transaction proposals are prepared by the client applications and then parsed to determine which parts of transaction deals with BC ledger and/or off-chain storage ( $S^s$ ). Basically, BC is document oriented, which contains nested sets of key-value pairs, whereas RDB is a relational key based structure. Although different, but RDB and BC have a commonality of a *key*. Based on this “key,” we use triples mechanism [33] that can effectively represent any kind of BC and RDB transactional data.

A *triple* consist of three parts, i.e.,  $S, P, O$  that express the subject, predicate, and object, respectively. Here, *subject* is the thing which helps in linking multiple triples, *predicate* indicates the relationship between subject and object, whereas *object* is a constant property of subject. For conversion purposes, we can define triples as  $\{S, P, O\} \leftrightarrow \{id, key, value\}$ . To better clarify the concept, assume an IoMT device for measuring blood pressure, where the observed value is 140. The triples for this will be  $\{id, key, value\} \rightarrow \{dev1, BP, 140\}$ .

For a more concrete running example pertinent to the scope of this article, consider the following BC transaction in JSON format:

$$T_i = \{id:12cf, path:e23fdæe234, pointer:123fec, file:[img1.jpg, img2.jpg]\}. \quad (1)$$

Based on this information, a relational database can have a record, as shown in Table II, where different pieces of information are stored as values under columns in a relation/table. Either of these formats can be restructured as triples. It can be seen in Table III that each element of the record/transaction can be presented in  $\{S, P, O\}$  format, where the ID can link multiple  $\{S, P, O\}$ .

As a higher level declarative language for specifying relational queries, the relational algebra shown in (2) depicts the migration of Table II into triple (see Table III). We use generic symbols for relational algebra syntax, which are also mentioned



TABLE I  
LIST OF SYMBOLS

Symbols	Description
$u_i \in U$	Any kind of User ( $SD_i, Pt_i, Ph_i \in U$ )
$P_i \in P$	A Peer in BC network
$P'$	Set of endorsing peers
$SD_i$	IoMT service device
$Pt_i$	Patient
$Ph_i$	Physician
$CA^{BC}$	Certificate Auth. of BC network
$CA^{CH}$	Certificate Auth. of CEHS
$S^a$	Application Server
$S^h$	Conventional e-health server
$S^s$	Off-chain storage server
$SC_i$	A smart contract
$T_i$	A single transaction
$T_i^{U_i}$	A transaction from user $i$
$B_i$	A block in ledger
$C_i$	A channel in BC network
$sk$	secret key
$pk$	public key
$sck$	secret compound key
$\sigma$	Select operation
$\pi$	Attributes
$\rho$	rename (table name)
$\bowtie$	Join operation

TABLE II  
TRANSACTION  $T_i$  IN RDB

id	pointer	path	file
12cf	123fec	e23fdae234	img1.jpeg, img2.jpeg

TABLE III  
TRIPLE REPRESENTATION OF TRANSACTION  $T_i$

id	key	value
$i_1$	id	12cf
$i_1$	path	e23fdae234
$i_1$	pointer	ae3224cfa
$i_1$	file	$i_2$
$i_2$	0	img1.jpeg
$i_2$	1	img2.jpeg

in Table I

$$\begin{aligned}
 T_i &= \rho_{T^s}(\text{id,pointer,path,file}) \\
 &\left( \pi_{(v_i, v_p, v_{pa}, v_f)} \left( \sigma_{(k_i=\text{id} \wedge k_p=\text{pointer} \wedge k_{pa}=\text{path} \wedge k_f=\text{file})} \right. \right. \\
 &\left. \left( \rho_{T_{\text{id}}^s(i, k_i, v_i)}(T^{tr}) \bowtie \rho_{T_{\text{pointer}}^s(i, k_p, v_p)}(T^{tr}) \bowtie \right. \right. \\
 &\left. \left. \rho_{T_{\text{path}}^s(i, k_{pa}, v_{pa})}(T^{tr}) \bowtie \rho_{T_{\text{file}}^s(i, k_f, v_f)}(T^{tr}) \right) \right). \quad (2)
 \end{aligned}$$

The equation explains a formal foundation for relational model operations, such as searching specific value triple where the same id presents multiple key values. This process also optimizes the queries for processing. Optimization modules through inner joins and their linkage results in information based on a single key. The same key is also used in BC ledger that helps in smooth synchronization. Fig. 4 shows the synchronization steps

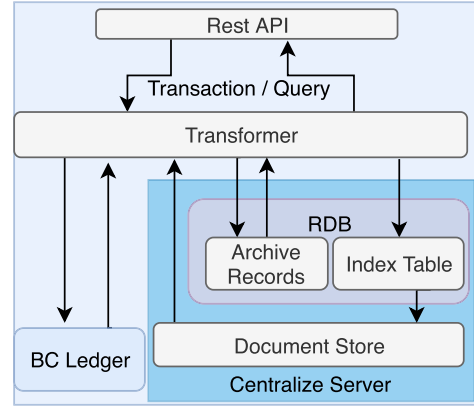


Fig. 4. Transaction/query transformation flow.

of transaction or query transformation in both systems. The transformer module is part of the migration DDC system and receives accept SQL/NoSQL syntax from the rest API of the users. It then forward it to BC ledger or the centralized server as per the information desired. If it is required to access off-chain storage  $S^s$  for heavy data, it has to collect reference points from the index table and then obtain the documents from  $S^s$ .

3) *Transformation of Transaction BC  $\leftrightarrow$  RDB*: As discussed earlier, heavy medical records, which are inserted through a BC transaction, are stored in off-chain storage while an RDB maintains the indices of such document/images. In response to a query, results are generated by the integration of data retrieved from BC ledger as well as RDB. Hence, it is crucial that automatic synchronization is maintained. According to the previous discussion, all RDB and ledger data can be presented as triples. However, because of structural constraints, it is impossible to use directly such triple data in the form of a hybrid database. For more clear understanding, (3) retrieves all medical images that have  $\text{id} = i_2$  as a table named  $T^p$  from triples shown in Table III. It is difficult to recognize and store such a nested piece of information, as it is not clear which data are associated with which user

$$\rho_{T^p}(\text{image}) \left( \pi_{\text{value}} \left( \sigma_{\text{key}=i_2} (T^{tr}) \right) \right). \quad (3)$$

To solve the issues, it can be presented as a set of key-value pairs, such as  $\{k_1 : v_1, k_2 : v_2, \dots, k_n : v_n\}$ . Moreover, each value can further have a nested key-value pair in the BC ledger as

$$\begin{aligned}
 \{k_i : [v_i^1, v_i^2, \dots, v_i^n]\} \equiv \\
 \{k_i : \{0 : v_i^1, 1 : v_i^2, \dots, n-1 : v_i^n\}\}. \quad (4)
 \end{aligned}$$

Here, each value  $v$  is represented as an *integerKey:value* within the value part of  $k_i$ .

The next challenge is to transform such nested key-value pairs into triples structure in BC ledger, which can be easily retrieved by SQL and NoSQL queries. For simplicity of understanding, consider (5) as an example, which is a nested structure of (1). For transforming a regular NoSQL transaction  $T_i^{ns}$  to a key-value pair, we define a function  $\lambda_i$  shown in (6), and  $\Theta_i$  shown in (7) for a complete set of key-value pairs. Finally, (9) shows the

complete conversion steps

$$T^{ns} = \{\text{id:12cf, path:e23fdae234, pointer:123fec, file:\{0 : img1.jpg, 1 : img2.jpg\}}\} \quad (5)$$

$$\lambda_i(T^{ns}) = \{(i, T_k^{ns}, T_v^{ns}), (i, T_k^{ns}, j) \cup \Theta_j(T_v^{ns})\} \quad (6)$$

$$\Theta_i(S) = \bigcup_{T^{ns} \in S} \lambda_i(T^{ns}). \quad (7)$$

Hence, we can write

$$\begin{aligned} \Theta_{i_1} &= \bigcup_{T^{ns} \in S} \lambda_{i_1}(T^{ns}) \\ &= \{\text{id:12cf, path:e23fdae234, pointer:123fec, file:\{0 : img1.jpg, 1 : img2.jpg\}}\} \quad (8) \end{aligned}$$

which can be expressed as

$$\begin{aligned} \Theta_{i_1} &= \bigcup_{T^{ns} \in S} \lambda_{i_1}(T^{ns}) \\ &= \{(i_1, \text{id}, 12\text{cf}), (i_1, \text{path}, \text{e23fdae234}), \\ &\quad (i_1, \text{pointer}, 123\text{fec}) \cup (i_1, \text{file}, i_2) \\ &\quad \cup \lambda_{i_2}(0 : \text{img1.jpg}, 1 : \text{img2.jpg})\} \\ &= \{(i_1, \text{id}, 12\text{cf}), (i_1, \text{path}, \text{e23fdae234}), \\ &\quad (i_1, \text{pointer}, 123\text{fec}), (i_1, \text{file}, i_2)\} \\ &\quad \bigcup_{T^{ns} \in \{0:\text{img1.jpg}, 1:\text{img2.jpg}\}} \lambda_{i_2}(T^{ns}) \\ &= \{(i_1, \text{id}, 12\text{cf}), (i_1, \text{path}, \text{e23fdae234}), \\ &\quad (i_1, \text{pointer}, 123\text{fec}), (i_1, \text{file}, i_2)\} \\ &\quad \cup \lambda_{i_2}(0 : \text{img1.jpg}) \cup \lambda_{i_2}(1 : \text{img2.jpg}) \\ &= \{(i_1, \text{id}, 12\text{cf}), (i_1, \text{path}, \text{e23fdae234}), \\ &\quad (i_1, \text{pointer}, 123\text{fec}), (i_1, \text{file}, i_2), \\ &\quad (i_2, 0, \text{img1.jpg}), (i_2, 1, \text{img2.jpg})\}. \quad (9) \end{aligned}$$

4) *Query Transformation*: In a BC, queries are executed from world-state, which is a NoSQL supported CouchDB. Hence, similar to the generic query, application (user interface) generated conditions are used as selectors in CouchDB, which in turn are expressed as a JSON object describing documents of interest. A selector is considered a *key*, which might be one or more fields, and the corresponding values required for those fields. Based upon the selector's data, whole database file is searched and response generated as a *value*. For example, according to Table III and (9), a

$$\{\text{selector} : \{\text{"id"} : \text{"12cf"}, \text{"file"} : \text{"i_2"}\}\}$$

matches whole database document with an *id* field containing *12cf*, and file *i\_2*, which will result in retrieval of respective images as values. Furthermore, more complex selector combining operators may be utilized for better performance based on requirements of the use cases.

As shown in Fig. 4, users generate transaction/query through REST API, which is executed and transformed in migration

---

### Algorithm 2: Query Processing.

---

```

Input : Query ( $T^{qry}, U_{id}, U_{sign}, SC^{ver}$ )
Output: Arg[]
1 if  $U_{id}, U_{sign}, SC^{ver}$  then
2   set  $key^{T^{qry}} \leftarrow \text{parse}(T^{qry})$ 
3   if  $key^{T^{qry}} \exists L$  then
4      $R^{T^{qry}} \leftarrow \text{Query}(key \exists L)$ 
5     if ( $\text{img.Hash} \leftarrow \text{parse}(R^{qry})$ ) then
6       set
7        $\text{img} \leftarrow \text{Query}(\text{Off-chain}, \forall_{key^{T^{qry}}})$ 
8       set  $R^{T^{qry}} \leftarrow \text{merge}(R^{T^{qry}}, \text{img})$ 
9     end
10    else
11      while  $U_{id}$  do
12        set  $R^{T^{qry}} \leftarrow \text{Query}(RDB(key^{T^{qry}}))$ 
13      end
14      Set  $R^{T^{qry}} \leftarrow \text{JSON.Conv}(Pt_{id} + \text{value})$ 
15    end
16   $R^{T^{qry}} \leftarrow \text{null}$ 
17 end
18 return  $R^{T^{qry}}$ 

```

---

framework, and the transformer module decides whether it will be executed on BC ledger or  $S^s$ , or both. This is done using Algorithm 2, which executes at the peer in response to the query from API. API executes the user's query statement with some parameters while the user's credentials are verified in line 1. In line 2, the main key is parsed from the query statement, and used for extracting information from BC ledger or RDB. Lines 3–8 searches in BC ledger, whereas lines 5–8 are used to retrieve images from off-chain storage. If the key does not exist in the ledger, it is considered to be available in RDB, which is executed by lines 10–14.

#### D. Memory Consumption Discussion

A national-level unified EHS generates big data that has to be secured in a variety of ways, which is a driving factor in migrating RDB to BC-based NoSQL DB. However, this does not mean that relational database systems are at loss. Although generic NoSQL overcomes the structural relational constraints and large dataset operations, transaction scaling and storage of heavy data are challenging BC. In this section, we determine the effect on memory consumption and how the solution mitigates excessive memory usage. As discussed earlier, and depicted in Fig. 3, each block in the chain stores transaction payloads along with metadata. It is important to note that in the proposed solution, heavy images/documents are not part of the transaction itself. This has two main reasons. First, as earlier mentioned, the block itself has size limitations, and hence cannot store them. Second and equally important is the fact that if large data are made part of the transaction, it will have to be transferred to each peer during the validation/consensus formation phase, and then again at commit phase. This will require huge bandwidth throughout the network, which makes it impractical. The following equations enable us to precisely measure the block

weights in our solution. Assume a group of users  $U_x$  invoke  $T_x$  transactions from their applications for a particular session where  $P'_n$  endorsing peers are involved, then the orderer creates a new block  $B_j$ . Equation (10) estimates the block weight in real-time where  $W$  denotes weight

$$W^{B_j} = \sum_{i=1}^x W^{T_i} + W(B_j^{\text{header}} + B_j^{\text{metadata}}) \quad (10)$$

where

$$W^{B_j^{\text{header}}} = W(B_{j-1}^{\text{Hash}} + \text{Hash}(\forall T_x)), \text{ and}$$

$$W^{T_i} = W\left(T_i^{\text{header}} + U_i^{\text{sign}} + T_i^{\text{data}} + \sum_{j=1}^n \left(P'_j(\text{resp}) + P'_j(\text{sign})\right)\right).$$

It is important to remember that transaction and block sizes have maximum limits, and our proposed scheme does not violate this restriction. The heavy data are not included in transactions, hence it does not impact the storage requirements in our proposed solution.

## V. SYSTEM PERFORMANCE EVALUATION

The evaluation of BC-based solutions is not trivial, as it requires not only a BC network but also the use case platform for which BC is being used. In order to evaluate the proposed synchronization system, we have carried out several experiments to evaluate different performance aspects.

### A. Testbed Setup

The overall system implementation is tested on the docker container based Hyperledger Fabric (v1.2) platform. We use two systems with the following specifications for emulating the topology: First, Intel i5 3 GHz processor with 8 GB of 1600 MHz DDR3 RAM, and second, Intel i7 2.7 GHz processor with 16 GB of 1600 MHz DDR3 RAM. The system models a four peer network each one implemented using docker containers and node-red based application is utilized to generate transactions of BC network, whereas Hyperledger composer based `cli` API supports the whole integration process. In the node-red environment, conventional e-healthcare network is developed where Fabric-in and Fabric-out nodes are used to act as CEHS and BC network. The bridge between nodes modulates the data flow to/from the CEHS and BC nodes. For evaluating existing centralized RDB for a CEHS, we have utilized MySQL DB in an emulated environment to store the off-chain image hash and paths, whereas IPFS has been used to store raw images. Hyperledger integrated CouchDB supports world state of BC network. The CEHS database is filled with synthetic data to mimic a large set of EMRs. As the objective of the evaluation is the efficiency of the BC network and conversion process, hence the emulated data are sufficient for this purpose. Furthermore, N1QL [34] is used as a JSON document model for generating query or transactions.

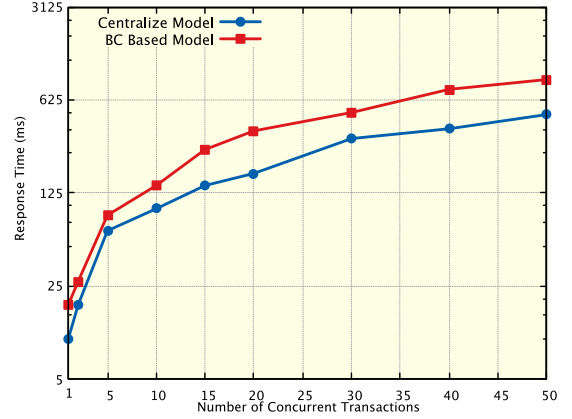


Fig. 5. Transaction completion time for CEHS and BC-EHS.

### B. General Observations

The complete unified system is based on a number of different types of transactions and processes. The simulation and docker log analysis result reflects the following general observation.

1)  $T_P$  Versus  $T_D$ : The system has two different types of transactions. Policy update transactions  $T_P$  are only initiated by patients that take approximately 10–18 ms. It should be clear that policy transactions are not regular transactions, and occur only when a patient changes the service providers. Moreover, they are added to the genesis block as a new version for the channel. Hence, they do not burden the peers or orderer in terms of transactions per second (TPS) or ledger size. The evaluations presented in the next section specifically deal with  $T_D$ .

2) *Key Generation*: For device registration and compound key generation, the proposed solution requires between 3–20 ms, which is quite small and does not impact the overall performance. Similarly, channel creation and permissions approval average is approximately 3 ms–1 s (depends on policy). It is important to note that key changes and channel version updates are not common for a given patient.

3) *Time Division*: The completion of a transaction (proposal to acknowledgment) requires  $\approx 3$  s – 180 s; however, the bulk of this time is spent on transaction preparation in the application, as compared to consensus formation or creation of the block. Hence, the performance of the BC network is not affected by it.

### C. Transaction Data Processing

In this set of experiments, we calculate the complete transaction  $T_D$  time, from creation by application to the final commit of a block. We compare it to the conventional CEHS solution, where only a single server performs all tasks, and a commit to RDB is the final step. Fig. 5 presents the transaction execution times, where we have fixed the number of endorsing peers  $P'$  to four. As a result, for a single transaction in BC, the unified EHS takes  $\approx 20$  ms, whereas a purely conventional system requires  $\approx 12$  ms. For five concurrent transactions, CEHS requires  $\approx 65$  ms, whereas BC-EHS requires  $\approx 85$  ms. Finally, for 50 transactions, the centralized model requires  $\approx 500$  ms, whereas BCEHS requires close to 900 ms. It can be observed that the increase in number of transaction steadily increases the response time in both systems. However, the BC system takes slightly



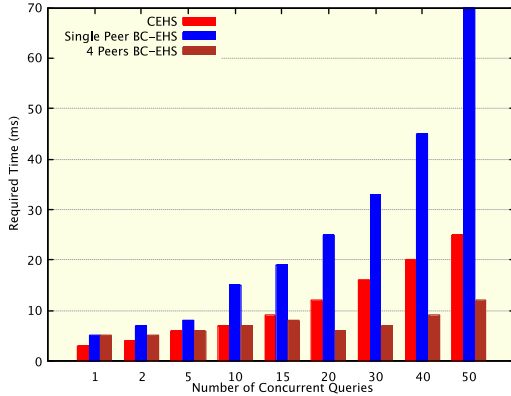


Fig. 6. Query time in CEHS, single/multiple peer BC-based system.

more time that is because each peer verifies the transaction, whereas in a CEHS, only a single server is responsible for it. This increase is not significant and in acceptable delay levels. As the scope of this article is not to improve the BC consensus formation time, hence we believe that as the BC technology matures, efficient algorithms will remove this difference in response times. An important factor to note is that TPS is highly dependent on the platform and its configuration. From our testbed, we observe that minor log captures can increase it by 20–32 ms on average.

#### D. Query Processing Evaluation

In contrast to the previous experiment, here, we evaluate the retrieval of data only, which is done through  $\tilde{T}_D$ . Due to the design of our solution, transactions are sent to peers rather than a single server, which can help in reducing the resolution time. Fig. 6 shows the impact of peer increase on TPS in comparison to the traditional CEHS. It is important to note that in the previous experiment, we measure the complete transaction time, whereas in this, we only measure the query processing time. It can be observed that as the number of concurrent queries is increasing, the processing time also increases. Although for a single query, BC and CEHS model times are very close ( $\approx 4$  ms), the difference significantly increases with increment in queries. For 50 concurrent queries, the centralized model requires 25 ms, whereas a single peer BC requires  $\approx 70$  ms. In a multipeer network, for 15 to 20 concurrent queries, the required time falls from approximately 8 to 6 ms. As queries are distributed to peers that are executed parallel, hence the query responses from these individual peers take less time. On the other hand, in a centralized system, every query response comes from a single server that requires more time.

#### E. Ledger Scalability

The main challenge of BC-based EHS is the storage of medical images/documents along with transactions. Based on our investigation, the size of a single medical image varies between 10 KB to 1 MB and in a real-life scenario, one transaction may have 1–10 such images. However, for this evaluation, we have limited the maximum size of an image to 1 Mb, due to testbed constraints. In the proposed solution, we have separated

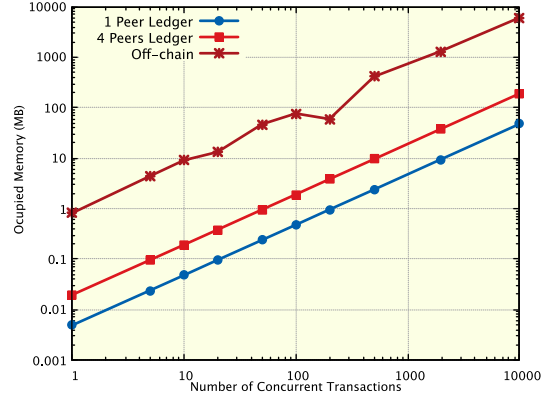


Fig. 7. Ledger expansion with medical image transactions.

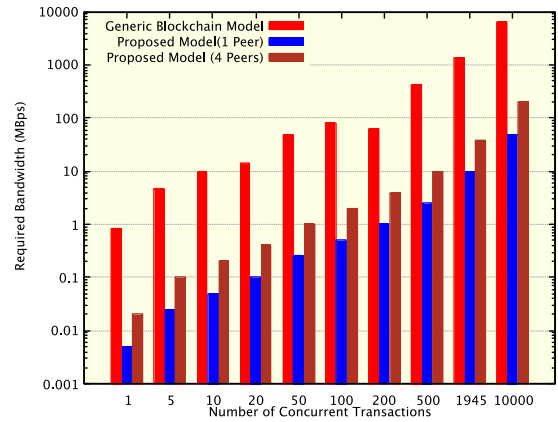


Fig. 8. Required bandwidth for consensus.

medical images from original BC transactions and stored them in off-chain storage. Fig. 7 depicts the memory required by a single ledger, all four ledgers, and the off-chain storage, against an increasing number of transactions for new data. The increase in memory space is almost linear and constant for that of a single peer, and as four peers maintain identical ledgers, hence their total memory requirement is also linear. The bulk of the heavy data is stored in the off-chain storage. This shows that the solution proposed in this article does not affect the efficiency of the BC itself. The ledger remains as scalable as in a nonheavy data environment, such as cryptocurrency transactions.

#### F. Scalability: Bandwidth Conservation

From a big data perspective, BC suffers from several scalability issues. These include the consensus process among  $n$  number of peers, the computational capacity of different elements, size of transactions and blocks, the memory capacity of the ledger, and bandwidth required to achieve desired TPS. Although in generic BC solutions, the consensus process can be customized; however validation of blocks or transactions must be performed by endorsing peers. This requires that the transaction be sent to them. Depending on the number and size of transactions, there should be enough bandwidth available to transfer them among nodes. Fig. 8 shows the required bandwidth using the proposed solution against a generic BC solution for heavy data trades  $\tilde{T}_D$ . It can be observed that if the heavy data are part

of 1K transactions, then the available bandwidth should be approximately 10 GB/s, to achieve the same efficiency (TPS) as that of our solution, which can work within 60 MB/s. Hence, the proposed solution not only reduces the memory requirements but also limits the network bandwidth required to create a unified BC-based EHS.

## VI. CONCLUSION

The number of public and private healthcare service providers has grown significantly in recent times due to the advancement in EHSs. Given their numerous benefits, they also suffer from challenges, such as sharing of information, national-level regulation and oversight, and security and privacy of information. The primary objective of this article was to use BC to provide a unified network of EHSs, where the complete ecosystem can share information and control access to it. It further addressed the management issue of merging the conventional and BC networks, and more specifically, the data storage in relational databases and file-based database structures. The proposed system first interconnected conventional e-health service providers to each other through a BC backbone for seamless exchange of patient data with strict access control. It then defined a unified data structure for data storage in different types of storage systems. Finally, it enabled the off-chain storage of information for large data that cannot be part of the chain. Implementation and analysis showed that not only the performance was within acceptable limits, but also strict user-defined access control policy eliminated unwanted data access in the system.

## REFERENCES

- [1] J. Worthington, "Global telehealth market set to expand tenfold by 2018," 2017. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.meddeviceonline.com/doc/global-telehealth-market-set-to-expand-tenfold-0001>.
- [2] P. Pace, G. Aloï, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 481–489, Jan. 2019.
- [3] Statista, Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions), 2016. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>
- [4] R. Scheffler, J. Liu, Y. Kinfu, and M. Poz, "Forecasting the global shortage of physicians: An economic and needs-based approach," Dec. 2018. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.who.int/bulletin/volumes/86/7/07-046474/en/>
- [5] C. Flanagan and S. Qadeer, "A type and effect system for atomicity," in *Proc. ACM Conf. Program. Lang. Des. Implementation*, 2003, pp. 338–349.
- [6] My Health Record: Control your health information securely in one place. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.myhealthrecord.gov.au/>
- [7] T. Y. Shu, "About the NEHR." Accessed: Apr. 19, 2020. [Online]. Available: [https://www.ihis.com.sg/Latest\\_News/Media\\_Releases/Documents/2018\\_%20Media%20Releases/Factsheet%20-%20NEHR%20\(2%20Jan\).pdf](https://www.ihis.com.sg/Latest_News/Media_Releases/Documents/2018_%20Media%20Releases/Factsheet%20-%20NEHR%20(2%20Jan).pdf)
- [8] A. Bahga and V. K. Madiseti, "A cloud-based approach for interoperable electronic health records (EHRs)," *IEEE J. Biomed. Health Informat.*, vol. 17, no. 5, pp. 894–906, Sep. 2013.
- [9] European Commission, "Overview of the national laws on electronic health records in the EU Member States and their interaction with the provision of cross-border eHealth services," Jul. 2014. Accessed: Apr. 19, 2020. [Online]. Available: [https://ec.europa.eu/health/sites/health/files/ehealth/docs/laws\\_report\\_recommendations\\_en.pdf](https://ec.europa.eu/health/sites/health/files/ehealth/docs/laws_report_recommendations_en.pdf)
- [10] A. Jain, "The 5 V's of big data," 2016. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/>
- [11] M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 4, pp. 3796–3838, Oct.–Dec. 2019.
- [12] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 1, pp. 858–880, Jan.–Mar. 2019.
- [13] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, pp. 61656–61669, 2019.
- [14] K. Zhang and H.-A. Jacobsen, "Towards dependable, scalable, and pervasive distributed ledgers with blockchains," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, Jul. 2018, pp. 1337–1346.
- [15] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Oct. 2017, pp. 1–5.
- [16] A. K. Das, P. Sharma, S. Chatterjee, and J. K. Sing, "A dynamic password-based user authentication scheme for hierarchical wireless sensor networks," *J. New. Comput. Appl.*, vol. 35, no. 5, pp. 1646–1656, Sep. 2012.
- [17] Q. Wang, Y. Zhu, and X. Luo, "Multi-user searchable encryption with coarser-grained access control without key sharing," in *Proc. Int. Conf. Cloud Comput. Big Data*, Nov. 2014, pp. 119–125.
- [18] Z. Yaling, J. Zhipeng, and W. Shangping, "A multi-user searchable symmetric encryption scheme for cloud storage system," in *Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst.*, Sep. 2013, pp. 815–820.
- [19] Z. Liu, Z. Wang, X. Cheng, C. Jia, and K. Yuan, "Multi-user searchable encryption with coarser-grained access control in hybrid cloud," in *Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol.*, Sep. 2013, pp. 249–255.
- [20] L. Yang, Q. Zheng, and X. Fan, "RSPP: A reliable, searchable and privacy-preserving e-healthcare system for cloud-assisted body area networks," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.
- [21] W. Liu, S. Zhu, T. Mundie, and U. Krieger, "Advanced block-chain architecture for e-health systems," in *Proc. IEEE 19th Int. Conf. e-Health Netw., Appl. Serv.*, Oct. 2017, pp. 1–6.
- [22] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 746–759, Apr. 2016.
- [23] T. van der Vorst, "Catena—SQL on a blockchain," Dec. 2017. Accessed: Apr. 19, 2020. [Online]. Available: <https://github.com/pixelspark/catena/blob/master/README.md>
- [24] Apache Sqoop Documentation. Accessed: Apr. 19, 2020, pp. 1–20. [Online]. Available: <https://sqoop.apache.org/docs/1.99.7/index.html>
- [25] DataX: Data Synchronization Tool. Accessed: Apr. 19, 2020. [Online]. Available: <https://github.com/alibaba/DataX>
- [26] G. Zhao, Q. Lin, L. Li, and Z. Li, "Schema conversion model of SQL database to NoSQL," in *Proc. 9th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.*, Nov. 2014, pp. 355–362.
- [27] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2343–2355, Mar. 2020.
- [28] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4650–4659, Jun. 2019.
- [29] L. Cocco, A. Pinna, and M. Marchesi, "Banking on blockchain: Costs savings thanks to the blockchain technology," *Future Internet*, vol. 9, no. 3, 2017.
- [30] G. Greenspan, "Scaling blockchains with off-chain data." Accessed: Apr. 19, 2020. [Online]. Available: <https://www.multichain.com/blog/2018/06/scaling-blockchains-off-chain-data/>
- [31] J. Worthington, "Storing images in Hyperledger fabric (blockchain)," Mar. 2017. Accessed: Apr. 19, 2020. [Online]. Available: <https://belltane.wordpress.com/2017/03/27/storing-images-in-hyperledger-fabric-blockchain/>
- [32] J. Han, H. E. G. Le, and J. Du, "Survey on NoSQL database," in *Proc. 6th Int. Conf. Pervasive Comput. Appl.*, Oct. 2011, pp. 363–366.
- [33] RDF Primer: W3C Recommendation. Accessed: Apr. 19, 2020. [Online]. Available: <https://www.w3.org/TR/rdf-primer/>
- [34] "CouchBase: Run your first N1QL query," Feb. 2004. Accessed: Apr. 19, 2020. [Online]. Available: <https://docs.couchbase.com/server/6.0/getting-started/try-a-query.html>



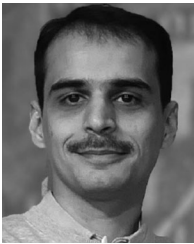
**Sujit Biswas** (Member, IEEE) received the M.Eng. degree in computer engineering from the Northwestern Polytechnical University, Xi'an, China, in 2015, and the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, Beijing, China, in 2020.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Faridpur Engineering College, University of Dhaka, Dhaka, Bangladesh. His current research interests include IoT, blockchain, mobile computing security and privacy, big data, machine learning, data-driven decision making, etc.



**Zohaib Latif** received the B.S. degree in electrical engineering and the M.S. degree in electrical and electronics engineering from the University of Glasgow, Glasgow, U.K., in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree in software defined networking with the School of Computer Science, Beijing Institute of Technology, Beijing, China.

Since 2011, he has been a Senior Lecturer with the School of Computer Science, Beijing Institute of Technology. His current research interests include software-defined networks (SDN), distributed controllers in SDN, and Internet of Things.



**Kashif Sharif** (Member, IEEE) received the M.S. degree in information technology from the National University of Sciences and Technology, Islamabad, Pakistan, in 2004, and the Ph.D. degree in computing and informatics from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2012.

He is currently an Associate Professor for research with the Beijing Institute of Technology, Beijing, China. His research interests include data-centric networks, blockchain and distributed ledger technologies, wireless and sensor networks, software-defined networks, and 5G vehicular and UAV networks.

Prof. Sharif currently serves as an Associate Editor for the IEEE ACCESS, and regularly serves on program committees of IEEE and ACM conferences.



**Salil S. Kanhere** (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from Drexel University, Philadelphia, PA, USA, in 2001 and 2003, respectively.

He is currently a Professor with UNSW Sydney, Sydney, NSW, Australia. His research interests include Internet of things, cyber-physical systems, blockchain, cybersecurity, and applied machine learning.

Prof. Kanhere serves as an Editor-in-Chief of the *Ad Hoc Networks* and is an Editorial Board Member of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *Pervasive and Mobile Computing*, and *Computer Communication*. He is an ACM Distinguished Speaker and a Humboldt Research Fellow. He regularly serves on the organizing and program committees of IEEE and ACM conferences. He is a Senior Member of the ACM.



**Saraju P. Mohanty** (Senior Member, IEEE) received the bachelor's (Hons.) degree in electrical engineering from the Orissa University of Agriculture and Technology, Bhubaneswar, India, in 1995, the master's degree in systems science and automation from the Indian Institute of Science, Bengaluru, India, in 1999, and the Ph.D. degree in computer science and engineering from the University of South Florida, Tampa, FL, USA, in 2003.

He is currently a Professor with the University of North Texas, Denton, TX, USA. His Google Scholar h-index is 35 and i10-index is 129 with 5400+ citations. He has more than 20 years of research experience on security and protection of media, hardware, and system. He introduced the Secure Digital Camera in 2004 with built-in security features designed using hardware-assisted security or security by design principle. He is widely credited as the designer for the first digital watermarking chip in 2004 and first the low-power digital watermarking chip in 2006. He has mentored two Postdoctoral Researchers, and supervised 11 Ph.D. dissertations and 26 M.S. theses. He has authored 300 research articles, four books, and invented four U.S. patents. His research is in "Smart Electronic Systems," which has been funded by the National Science Foundations, Semiconductor Research Corporation, U.S. Air Force, IUSSTF, and Mission Innovation.

Prof. Mohanty was a recipient of 12 Best Paper Awards, including the IEEE Consumer Electronics Society Outstanding Service Award in 2020, the IEEE-CS-TCVLSI Distinguished Leadership Award in 2018, and the PROSE Award for Best Textbook in Physical Sciences and Mathematics category from the Association of American Publishers in 2016 for his *Mixed-Signal System Design* book published by McGraw-Hill. He has delivered nine keynotes and served on five panels at various international conferences. He has been serving on the Editorial Board of several peer-reviewed international journals, including IEEE TRANSACTIONS ON CONSUMER ELECTRONICS and IEEE TRANSACTIONS ON BIG DATA. He is the Editor-in-Chief for the *IEEE Consumer Electronics Magazine*.



**Fan Li** (Member, IEEE) received the B.Eng. and M.Eng. degrees in communications and information system from the Huazhong University of Science and Technology, Wuhan, China, in 1998 and 2001, respectively, the M.Eng. degree in electrical engineering from the University of Delaware, Newark, DE, USA, in 2004, and the Ph.D. degree in computer science from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2008.

She is currently a Professor with the School of Computer Science, Beijing Institute of Technology, Beijing, China. She has more than 100 publications in reputed journals and conferences. Her current research interests include wireless networks, ad hoc and sensor networks, and mobile computing.

Prof. Li was the recipient of Best Paper Awards for her papers from the IEEE International Conference on Mobile Ad-Hoc and Smart Systems in 2013, IEEE International Performance Computing and Communications Conference in 2013, ACM International Symposium on Mobile Ad Hoc Networking and Computing in 2014, and Tsinghua Science and Technology in 2015. She is a Member of the Association for Computing Machinery.