applied sciences

MDPI

Article

Integrity and Privacy-Aware, Patient-Centric Health Record Access Control Framework Using a Blockchain

Rayan Anwar Abutaleb *, Saad Said Alqahtany and Toqeer Ali Syed

Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia
* Correspondence: rayan.abutaleb@psafhm.med.sa

Abstract: Blockchains are gaining traction as secure and reliable platforms for data sharing in fields such as banking, supply chain management, food production, energy, the Internet, and medical services. Furthermore, when decentralized, a blockchain can be regarded as an immutable ledger storing data entries. Moreover, this modern technology was designed to disrupt various data-driven industries, including the healthcare industry. While electronic healthcare services have enabled more straightforward and accessible treatment, patient privacy has become vulnerable to external and internal attacks by healthcare personnel. Therefore, we aimed to design a framework to control patient health records that ensures the patient can provide the necessary permissions to those who access his/her health records. This framework will record all activities via blockchain and usage control. Through this framework, we aim to create a user-centric and privacy-aware experience. A literature review and experiments have been performed to select an optimized and placable blockchain operating system. In addition, performance analysis showed that the OS and smart contracts work at an acceptable speed.

Keywords: health record; blockchain; Hyperledger Fabric; usage control; privacy

Citation: Abutaleb, R.A.; Alqahtany, S.S.; Syed, T.A. Integrity and Privacy-Aware, Patient-Centric Health Record Access Control Framework Using a Blockchain. *Appl. Sci.* **2023**, *13*, 1028. https://doi.org/10.3390/app13021028

Academic Editor: Gianluca Lax

Received: 13 December 2022
Revised: 2 January 2023
Accepted: 10 January 2023
Published: 12 January 2023

1. Introduction

A blockchain is distributed ledger technology (DLT) that allows data to be securely recorded across a network of peer nodes in geographically different locations. The integrity of data is an important factor in a health record. Similarly, the privacy of data is a great concern. One can store it with a third-party system without having control or monitoring [1]. Blockchain provides integrity and trust out-of-the box by using consensus algorithms. However, privacy and monitoring of data are out of the scope of blockchain technology. In this research, we have provided a complete integrated solution that helps patients with the integrity, privacy and monitoring of data with smart contracts and usage control.

Though the blockchain emerged with the creation of Bitcoin, the scope of blockchain research has expanded to many areas, including healthcare. The third generation of blockchain technology (BT) is now concerned with non-financial applications [2]. BT is a novel decentralized infrastructure and distributed computing model that receives the data and stores them securely in a database. The data stored in the system are integrity-aware. That means if any tempering happens to the data, it will be detected. This capability of blockchain motivated us to use it for patient health record systems. However, sometimes it is necessary to retrieve and update the data.

This features means we have to provide control of the health record to the patient. Thus, if anyone would like to retrieve a health record, it will be with the permission of the owner (patient). If any unauthorized tempering with the health record happens, it will be detected and reported to the patient. Now providing access to the health record in the existing system is static. Once you provide access to the data, it is usually provided for an unlimited time. You cannot monitor the access for a specific time and then revoke it. To avoid this, we provide a usage control model in this research which provides access to

the owner (patient) so he/she can grant permission to the specific health personnel for a specific period of time.

The blockchain provides a trusted mechanism with the help of a consensus algorithm [3,4]. Some consensus algorithms work on truly anonymous data, such as proof-of-work (PoW). However, the problem with PoW is that it is an order-executing architecture. That means the transaction is generated and then waits for the specific hash to be generated by minors. This process takes plenty of time. On average, it takes five to ten minutes. Due to this problem, we searched for permissioned blockchain systems which efficiently execute the transactions compared to public blockchains. We selected the Hyperledger Fabric blockchain operating system after the analysis phase of our research. Further details of the blockchain terminology, such as decentralization, immutability, and transparency, are explained in Section 3.

Therefore, we aimed to design a framework for controlling patient health records that enables patients to provide the necessary permission to those who will access their health records. This framework will record all activities via blockchain and usage control (UCON). Through this framework, we aim to create an experience that is user-centric, private, and secure.

The significant contribution of this new healthcare record framework is to provide a user-centric, privacy-preserving health record system with a blockchain, which also includes efficient solution selection and its criteria, and the complete process discussed in this paper. Any health record access is continuously monitored by the usage control model, which is the novelty in this work. Patients can establish parameters around the time frame in which healthcare records are available to concerned health personnel.

2. Research Objectives

The current research seeks to enhance the security and privacy of patients' data in hospitals using BT. Furthermore, the following goals will be achieved:

- To investigate the existing blockchain frameworks and select the optimized system in terms of performance and security.
- To design a framework that preserves the privacy and security of patients' records in a healthcare-record system.
- To implement and test user-centric privacy preserved by a secure HRS system on top of the blockchain framework.

3. Background

Our research's first objective was to find an optimized blockchain operating system. We have two types of blockchain: public and permissionless. Public blockchains include Bitcoin, Ethereum, ConsenSys Codifi, and EOS. Permissionless blockchains include Hyperledger Fabric, Ripple, and Corda. Initially, for our solution to this problem, we did some analysis, and we found different types of blockchain operating systems and evaluated them. Based on that, we have explained the different types of blockchains, and then we explain the components of blockchains.

Blockchain is a decentralized and distributed database of data protected by cryptographic techniques. Stakeholders (e.g., patients, payers, and other third parties) should upload data to the chain in a secure and verified manner. The result is a complete medical health record that can only be accessed by individuals with patient authorization, which smart contracts enforce. Once trust is established, participants merely need to communicate with the blockchain, which utilizes accepted interoperability standards (for example, FHIR). Smart contracts are independent transactions executed when specified terms of an agreement are met. The creation of contracts originated for the sake of establishing trust between two parties. Thus, all the data are safely shared. Rather than relying upon multiple points of connection, file templates, and exchange protocols (all of which pose a security risk and can be costly to address), a universally accessible blockchain reduces the total risk

to the participating entity while simultaneously enriching information sharing and patient involvement [5].

3.1. Blockchain

In a study by Nakamoto [6], BT was initially described in the form of Bitcoin. Blockchain is a distributed digital storage platform that captures transaction details in a ledger and stores them across several nodes in a transparent manner [7,8]. Blockchain is also known as a public ledger of transactions because all participants in the transaction disseminate and disclose the data in a bundle rather than retaining the record of transactions only on the central server, as is the current technique in data transactions [9].

A blockchain is a chronological series of blocks that includes a list of complete and valid transaction data. A reference (hash value) links each block to the one before it, producing a blockchain. The parent block is the block that comes before a specific block, and the genesis block is the first block [10]. A blockchain functions as an immutable timestamp ledger of blocks that can be shared between all participants in the network, so there is no need for a central authority [11]. Peer-to-peer networks employ this technology to distribute and store data in a distributed manner [12]. A blockchain is helpful for financial transactions and in various other situations. Decentralized IoT includes identity-based PKI, a decentralized supply chain, decentralized evidence of document existence, decentralized storage, and so on [13]. BT can be applied to various non-financial industries, including IoT, healthcare, education, and a variety of other non-financial start-ups [14].

All data are stored in blocks in the blockchain [15]. However, when a large amount of data is saved in blocks, the entire network's pace is reduced, personal information cannot be recorded, entered data cannot be removed, and all data are visible to all participants—a privacy infringement that violates the Personal Information Protection Act [16].

Using on-chain and off-chain concepts, blockchain overcomes this problem [17]. All network actions recorded on the blockchain are referred to as on-chain [18], and all actions that occur outside of the blockchain are referred to as off-chain [19]. This method can alleviate the decreased pace and the challenge of protecting sensitive information, such as personal data, because it does not entail writing directly to the blockchain [20]. Furthermore, data saved off-chain are validated on-chain to assure data forgery is prevented [21].

Blockchain's unique design is based on three key components: cryptography, peer-to-peer networks, and consensus mechanisms. These three characteristics provide the blockchain a high level of power, allowing it to be used in various applications (not just for digital cash). The following is a list of blockchain technology's features [22]:

- Decentralization: The blockchain's information is copied and dispersed by the network's nodes, which can independently validate the blockchain without needing a central authority.
- Immutability (tamper-proof): The neighbors keep a permanent record of transactions (ledger). As a block is cryptographically locked in the ledger once it is upended, it cannot be updated. This instills confidence in the transaction record.
- Transparency: The ledger contains a complete record of all transactions, and anyone can examine and audit transactions on the blockchain because it is an open file. This provides provenance, allowing asset lifetimes to be traced.
- Time-stamped: Blockchain's cryptographic technique connects blocks in a chronological chain, allowing for a trial of the underlying transactions. Cryptography in blockchain ensures the network's security [23]. All blocks of the blockchain are linked to each other so that each block is linked to the one before it and the next one, making it difficult for an attacker to edit any record. Doing so would also require changing the records or blocks related to manipulated records. This is nearly impossible in an extensive network with many blocks in a blockchain [24].

3.1.1. Types of Blockchain

The access control method used in a blockchain determines the type of blockchain. Anyone can join and participate in a public or permissionless blockchain, whereas private or permissioned blockchains restrict access [25]. Semi-permissioned blockchains allow everyone accesses to the network but confine certain activities (e.g., writing) to a group of individuals [26]. A separate category of both was defined by researchers who argued that both networks would be feasible. Blockchain network information was also collected in the assessment to provide a more granular view. The term "blockchain network" refers to a specific use of the BT (e.g., Bitcoin, Ethereum, and Hyperledger). Several authors have presented a novel blockchain network based on existing networks or using BT rather than using current blockchain networks. BTs can be categorized into three primary types based on how they are employed in different application situations [27,28]. These types are compared and characterized in Table 1.

- A public blockchain provides an open platform for individuals from various organizations and backgrounds to join, transact, and mine, in addition to reading and writing on the blockchain. All these characteristics are unrestricted, and anybody can send transactions, keep a copy of the distributed ledger, and participate in verifying and adding new blocks to the chain where permissionless blockchains are nominated [28].
- Private blockchains: This type of blockchain is primarily designed to enable private sharing and data exchange amongst a small number of known people inside a single company. Permissioned blockchains are similar to private blockchains in that external users cannot access or participate unless they have been granted permission [29]. Unlike public blockchains, write access is limited.
- Consortium blockchains: This type of blockchain can be considered a partially private and permissioned blockchain in which the consensus process and block validation are handled by a group of pre-selected nodes rather than a single entity. Due to the control of some selected validator nodes, a consortium blockchain comprises an extensive, centralized system. This type of blockchain is similar to private blockchains in that there are no processing fees and publishing new blocks is not computationally expensive [27]. Table 1 compares these blockchains.

Table 1. Types of blockchains [29].

	Public Blockchain	**Consortium Blockchain**	**Private Blockchain**
Registration authorities	Anyone	Multiple entities (organization)	Single entity defined before initialising the network
Access	Public read/write	Can be restricted	Can be restricted
Identity	Pseudo -anonymous	Approved participant	Approved participant
Immutability	Nearly impossible to tamper	Could be tampered	Could be tampered
Participation in consensus	All node	Selected node in multiple organisation	Single organisation
Transaction speed	Slow	Lighter and faster	Lighter and faster

Due to the nature and objective of their research or implementation, most of the original articles (see Table 2 below) chose a public permissionless blockchain. In the research, Ethereum (a public, permissionless blockchain network) provided a decentralized method of establishing an oracle [30–34] and a data feed service for smart contracts [35,36]. Nonetheless, certain decentralized oracle network protocols [37,38] and prediction market

platforms [39] have chosen to build their blockchain networks with logic tailored to their needs and objectives. For example, custom-built blockchain networks are used by the developers of self-verifying RFID chips [40] and cloud-based drone systems [41] to meet the needs of their particular solutions. Early researchers used WaltonChain, and consequent blockchain developers used DroneChain. WaltonChain and DroneChain were created to showcase a supply chain monitoring system and a resilient IoT architecture. Researchers at the University of Constanza [42] used OriginStamp, a web-based, trusted timestamping service that uses a decentralized blockchain to store anonymous, tamper-proof timestamps of any digital content. OriginStamp captures product meta-information (e.g., location, temperature, noise, or acceleration) collected by a smartphone. Users can select the type of blockchain on which the hashed timestamp will be stored using this web API.

Permissioned blockchains are also mentioned in publications. EdgeChain [43], a blockchain-based edge IoT framework, and EdenChain [44], a programmable economy platform, created solutions on top of a private Ethereum blockchain; the project owners built their access method and environment utilising Ethereum's technological stack. Datapace [45], a blockchain-based decentralized data marketplace, used Hyperledger Fabric to secure digital asset management, a vital component of their business. In another study [46], a novel network dubbed SpeedyChain was established to support the sophisticated process underpinning vehicle blockchain networks. Other researchers [47–49] have suggested employing a proprietary blockchain solution without providing network information. The creators of blockchain-based identity management [50] and cloud-based device commissioning [26] projects recommended employing both or semi-permission blockchains, naming their bespoke blockchain networks BlockID and ChainAnchor, respectively.

Table 2. Blockchain permissions.

Blockchain Type	Blockchain Network	Papers
Public permissionless	Ethereum	[30,31,34–36,43,44]
	Witnet	[37]
	Aeternity	[38]
	Truthcoin	[39]
	Waltonchain	[40]
	Dronechain	[41]
	OriginStamp	[42]
Permissioned	Hyperledger Fabric	[45]
	SpeedyChain	[46]
	Not Available	[47–49]
	BlockID	[50]
Both	ChainAnchor	[26]

3.1.2. Blockchain and Network

Mobile edge computing (MEC) integrates mobile access networks and Internet services by sinks computing to the edge of networks [51]. Therefore, privacy protection here is essential, which prompted the authors to use the blockchain with heterogeneous MEC. They proposed a (BlockTC) to solve routing trust and privacy problems. The BlockTC will provide a solution of trusted routing for cross-domain routing by using backup dual links and an accommodative bloom filter (ABF). The authors note that the combination of blockchain and distributed multi-domain networks can protect privacy for MEC systems.

Industrial manufacturing is developing through the integration of the Internet of Things (IoT) and industry, which led the industry of the Internet of Things (IIoT) to provide a general interconnection system [52]. Thus, it is difficult to protect privacy due to the

large number and types of terminals. In addition, the digital identity and data of access devices are semi-transparent to devices in the network, and this is an issue that must be taken care of in the IoT. The Authors proposed novel, private, blockchain-enabled trusted anonymous access (BlockTrust); a blockchain-enabled, tripartite, anonymous-identification trusted-service provision scheme (TriTrustServ) has been proposed to guarantee a balanced tradeoff among credibility, confidentiality, and efficiency. Their results show that using blockchain in their proposal can address anonymous trusted access.

Software-defined optical networking (SDON) control plane security is essential, especially in the network. The authors have proposed a distributed control architecture for SDON using the blockchain technique (BlockCtrl) to address the security [53]. Their results show that BlockCtrl can detect attacks such as DoS attacks and fault-tolerant control and provides high performance.

3.2. Blocks

A block's contents can be divided into the header and the payload, which contains a set of transactions. A Merkle tree is often built from the payload's transaction chain, and a Merkle root can be calculated using this Merkle tree. The Merkle root is used to create the block's hash, which is included in the header. The header also includes a hash pointer to the previous block and the date when the block was produced [6]. As the Merkle root becomes the block's hash, changing one transaction within a block is enough to modify the block's hash [54].

3.3. Nodes

A blockchain network comprises multiple nodes that are ideally connected peer-to-peer. Without a centralized authority, transactions between nodes occur directly amongst the nodes concerned. Each node in the network typically keeps a local copy of the whole blockchain, and as a result, determining the state of the blockchain in all nodes requires evaluating a single node inside the network. Moreover, since hierarchy does not exist, all nodes are treated equally [55]. However, distinct roles may exist among the nodes that make up the network; nodes can perform the tasks of a complete blockchain node or a miner node. A lightweight node is used in some blockchain implementations, including Bitcoin [29].

3.4. Creation of New Transactions and Blocks

When a transaction on a node in the network is finished, the node tells other nodes in the network about the transaction. A list of temporary transactions, called a transaction or memory pool, is also kept and maintained by nearly every node in the network. When the contents of a node's transaction pool are updated, they are transmitted to its nearby nodes [56]. However, a network's nodes do not need to form a complete network; all nodes do not have to be connected directly. Therefore, transmitting a given node's transaction list to all nodes in the network may require many steps. If a node receives transaction information that contradicts a transaction already stored in the specified node's transaction pool, the additional information is deleted [56].

A disagreement may occur if two contradictory transactions occur simultaneously at different nodes in the network. Both nodes will make every effort to inform as many other nodes as possible about their transaction, and the other nodes will regard the first valid transaction they receive as the correct one. However, network nodes must agree on a single blockchain state. Therefore, miner nodes must choose which transactions should be included in the next block. Miner nodes, like full nodes, keep track of all incoming transactions in their transaction pool [55].

3.5. Cryptographic Hash Functions

A hash function can take an amount of data as input and outputs a string with a specified number of bits. The input data are a message, and the output text is called a

hash [57]. Hash functions are used in various information security mechanisms, including digital signatures and password protection [58]. A hash function can have a variety of characteristics, including the following:

1. Deterministic: Given an input M, the hash function H returns a unique output H(M) = x [57].
2. One-way: Given an input M, computing H(M) = x is simple. However, other than brute force, it is impossible to extract M from H(M) = x given x [56].
3. Collision-free: It is infeasible to find two values for M and M' such that M6 = M0 and H(M) = H(M0) [56].

A hash function must satisfy all three properties listed above to be considered usable in cryptography [59].

3.6. Merkle Tree

A Merkle tree [60] is a tree-like data structure known as a hash tree. Every leaf node in a Merkle tree has a hash value. The hashes of the non-leaf nodes in the tree are computed by adding the hashes of the node's two children. When traveling one layer above the tree, the number of hashes will be reduced by half. Moreover, because the hash value of the root node depends on every other node in the tree, the hash value can be used to describe the complete tree [61].

3.7. Digital Signatures

Besides hash functions, digital signatures are an unavoidable cryptographic primitive in blockchains. These primitives generally provide source authentication, non-repudiation, and integrity. Typically, a digital signature system consists of two algorithms: the first one for signature generation and the second for verification. The produced signature largely relies upon a secret signature key possessed by the signer. The verification step uses a public key to validate the received signature. ECDSA and EdDSA are two digital signature methods commonly used in blockchains [62], and both are based on the elliptic curve variant of the discrete logarithm problem's hardness [29].

3.8. Consensus Procedures

A consensus protocol is a method for disseminating requests among nodes so that each node executes an identical set of requests on its instance of the service [63]. A blockchain uses a consensus method to determine the truthfulness and timeliness of blocks under consideration; blocks are accepted or refused depending on whether a consensus has been reached. Several consensus algorithms exist, including PoW [6], proof of stake [64], proof of elapsed time, and Kafka. A description of each is beyond the scope of this paper, so we refer the reader to the references provided for additional exploration [5].

Permissionless and permissioned blockchain solutions can be distinguished depending on the membership mechanism (i.e., how the identification of the user and their right to participate in the consensus are determined inside a network, such as a PoW or an endorsement policy) [65].

3.9. Smart Contract

Smart contracts are short programs on the blockchain that run business logic in reaction to changes in the blockchain or network architecture [25]. Smart contracts are available on almost every blockchain platform. The original goal of a smart contract was to reflect traditional written contracts, eliminating the need for trusted third parties, such as lawyers, by ensuring that contract parameters were met [25]. A smart contract can be any program that executes business logic and is suitable for deployment on a blockchain.

A smart contract often provides a set of conditions that must be met for the program to run. A smart contract takes a set of inputs and produces an output as a transaction, as shown in Figure 1. These limits cannot be altered when the software is placed on the immutable blockchain [25].

Figure 1. A program executing by smart contract (takes input and produces matching) [66].

3.10. Hyperledger Fabric–Chainlink

3.10.1. Fabric Network Architecture

Hyperledger Fabric [67,68], an open-source blockchain endeavour sponsored by the Linux Foundation, is an implementation of a permissioned blockchain. A security architecture for authentication and permission is built into Hyperledger Fabric (membership service (MS), employing a certificate authority (CA), which is an entity that can generate certificates for key pairs for signing and encryption for the peer nodes and solution users (SUs)). An MS's objective is to enable peer and user enrolment and transaction authorization using public-key certificates. This is one of the fundamental distinctions between permissionless blockchain architecture and the permissioned blockchain framework. Hyperledger Fabric also supports anonymous credentials with multiple CAs and the usage of threshold signatures. Besides the MS, peers and an ordering-service node, or orderer, are the other primary architectural components of Hyperledger Fabric. The orderer is a node (or a group of nodes) that runs the communication service that ensures delivery, such as atomic or complete order broadcast. Transaction verification and order are used to accomplish this. The digital signature of the transaction issuer and the so-called endorsement policy are validated during the authentication process. The endorsement policy is a specification for a chain code that tells a peer how to determine whether a transaction is valid. For example, all network peers must validate (and so sign) a transaction. The orderer must verify that all peers have signed the transaction and that the signatures are legitimate [18].

3.10.2. Types of Peers

Peers can play a variety of functions in Hyperledger Fabric [69]:

- A committing peer is in charge of committing validated transaction blocks in blocks. Every peer in a channel who has a copy of the ledger is a committing peer.
- An endorsing peer is specified by policy as a specialized node that simulates smart contract operations and responds to the client application with a proposal answer (endorsement).
- Defining a smart contract's endorsement policy entails identifying the organizations whose peers have to digitally sign a transaction before committing to the ledger.

3.10.3. Identity and Scalability

Blockchain networks can be found in orders, peers, administrators, client applications, and other nodes. Every actor must have a digital identity such as an X.509 certificate. Every actor in the blockchain network has permissions for resources defined by their identities.

3.11. Usage Control (UCON)

3.11.1. UCON Model

UCON is a conceptual structure that systematically covers these topics to create a general-purpose, unified framework for securing digital resources. UCON does not replace traditional access control, trust management, and digital rights management. However, UCON incorporates all three of these domains and extends beyond them in terms of definition and scope. Furthermore, even after the entities have been spread, UCON maintains fine-grained control over digital resources [70]. UCON encompasses obligations, conditions, continuity (ongoing controls), and mutability in the context of access control. Traditionally, access control has been limited to authorising a subject's access to specific

resources. Obligations are the conditions the subject must meet for access to be granted. Conditions are environmental criteria that must be met for access, regardless of the subject or object. Obligations and conditions are essential decision elements for more affluent and more nuanced restrictions on the use of digital resources in today's highly dynamic, distributed world [71].

Park and Sandhu proposed UCON that accommodates diverse and dynamic environment conditions [72]. Their attribute-based access control considers three variables when making an access decision: authorization, obligations, and conditions. Furthermore, attribute mutability (updates) and access decision continuity are two significant aspects that distinguish their UCON model from other traditional access control methods. The UCON model maintains a user session in three phases: pre-, ongoing, and post-access.

As UCON is an attribute-based access-control model, it can fulfil security requirements by incorporating several decision criteria, which makes it more reliable and versatile [73]. This approach primarily restricts the use of digital items while allowing classic access-control models to be included. Previous access-control models solely considered permission rules, and the UCON model also considers commitments and environmental conditions.

Consumers, providers, and identifiers are the three subjects identified by the UCON paradigm. Consumers are the people who ask for an object to execute a specific action. Individuals who own services and grant rights to the requesting party are referred to as providers. The identifier is the person or entity whose personal information is stored in a digital object. It is an optional collection of subjects that may or may not be present, depending on system requirements. However, it is always present if the system contains sensitive user information. Three categories of rights (actions) are given based on the employment activities of subjects: consumers, providers, and identifiers. Their rights indicate the set of actions or privileges on digital objects [72]. Other actions that fall under the category of performing adjustments in attribute values during the stages of a usage session are referred to as UCON actions [74].

3.11.2. Authorization Models of UCON

UCON pre-authorization models follow the same procedures as traditional authorization models. Before providing authorization for the requested resource, these models confirm user credentials and resource properties. Pre-authorization models can have immutable and changeable attributes (pre, ongoing, and post updates); thus, the values of the attributes change before, during, and after the access phase. Continuous verification is performed through ongoing authorization models, which analyze attribute values while the user is using the resource. As with pre-authorization models, ongoing authorization models can have both immutable and modifiable properties. As of the continual authorization evaluation in the access-control model, access permissions may be withdrawn throughout a session if a given attribute value changes [75].

3.11.3. Obligation Models of UCON

Obligation models cover the pre–post-obligation monitoring of access requests. These models enhance the decisions by requiring the user to complete the access-related obligatory tasks before being granted access. Pre and ongoing responsibilities refer to obligations used before and during usage. Post obligations are used to perform tasks after an access session has ended, such as sending access fulfillment alerts to the service provider. These post responsibilities can influence future usage session decisions by triggering policy update requests to the policy repository [76]. Thus, obligation checks are performed before, during, and after the access phase using respecting obligation models with immutable and mutable characteristics.

3.11.4. Condition Models of UCON

To accommodate environmental restrictions and system-related characteristics, UCON provides two condition models: pre and ongoing conditions with immutable attributes.

Conditions are reviewed the same way as authorization rules before and during the session. However, condition models ignore mutable features, such as a subject's shifting position [75].

4. Related Work

4.1. Blockchain in Healthcare

Notably, implementing blockchains in healthcare is recommended to deconstruct information-sharing barriers inherent in disparate, siloed EHR systems; to empower patients through data consolidation and access controls (e.g., secure and verifiable permissions, form completion, discharge instruction review, and patient-generated data contribution); to enhance the quality of healthcare while decrease costs and fraud; and to enhance data integrity, validation, and provenance [77–79]. Most prior studies have been proofs of concept or pilots to address the issues of information security, interoperability, data integrity, identity validation, and scalability that have stymied acceptance of this technology [66,79].

Recently, researchers have used BT to address security and privacy issues with healthcare data and recommend new strategies [80,81]. For example, Zyskind and Nathan [82] employed BT to create a personal data access control and management platform. They concentrated on users' privacy. Their platform enables this by combining a blockchain, re-purposed as an access-control moderator, with an off-blockchain storage solution while maintaining links to these data on the blockchain. When the user registers for the first time, a new shared identity is generated and sent to the blockchain. The data collected are encrypted using a shared encryption key and also sent to the blockchain, which subsequently routes it to an off-blockchain key-value store.

Meisami, Beheshti-Atashgah, and Aref [13] suggested a blockchain-based protocol for e-health systems that do not require third-party trust and provide an effective privacy-preserving authentication protocol. Unlike Bitcoin, transactions in their proposed model were not purely financial, and they did not employ traditional blockchain consensus techniques, such as proof-of-work (PoW), to achieve consensus. As the Internet of Things devices (IoT) have resource limits, it is incompatible with IoT applications. Utilizing proper consensus approaches improve network security and efficiency while lowering network costs, such as bandwidth and CPU utilization.

To improve privacy and data security across healthcare applications, Liu, Crespo, and Martnez [1] presented a blockchain and a Distributed Ledger-based Integrated Biomedical Security System (BDL-IBS). As it involves controlling and accessing a large amount of medical information, this technique can preserve data to ensure reliability while allowing patients to use data to support their care and providing robust consent systems for sharing data between different institutions and systems. Their findings revealed that emerging blockchain-based digital platforms provide rapid, easy, and seamless interactions amongst data suppliers, thereby enhancing privacy and data security for all stakeholders, especially patients.

Lee, Chang, and Kung [83] suggested a medical information preservation strategy that considers the complete data storage process of devices, from wearables to mobile phones to medical center servers. The entire procedure was secure and adhered to HIPAA privacy and security guidelines. The suggested approach utilizes extended chaotic map technology to create ID-based key negotiation for wearable devices, thereby lowering the amount of processing that wearable devices must perform and achieving lightweight quantification. Furthermore, the approach utilizes blockchain's non-tampering capability to ensure that data are not tampered with, boosting data security. The suggested approach can withstand attacks and is computationally lighter than previous methods, such as elliptic curve point multiplication, while maintaining security.

Sharma and Balamurugan [24] presented a blockchain-based approach for implementing EHRs and making them safer and more private. With its encryption protocols and decentralization, BT enables control over information access, striking a balance between

data privacy and data access. This project's major goal was to provide a framework for addressing data privacy and security challenges in electronic health care.

Ivan [84] proposed a new methodology for encrypting health data using a public blockchain (a decentralized database system with open access control for everyone connected to the network). The encrypted healthcare data are kept openly in this approach, resulting in the development of a blockchain-based personal health record (PHR). Ivan's proposed method provides patients with improved access to their clinical data, allowing them to freely access, monitor, and add to their records while sharing them with any associated caregiving agency.

In another study, Chen et al. [85] suggested an integrated blockchain and cloud-storage-based framework for managing and sharing patient personal medical data. The proposed approach could be used to store and communicate the personal medical data of patients safely and securely. The proposed approach is unique in allowing individuals total access to and control over their medical data, obviating the need for external third-party engagement.

4.2. Healthcare Systems

An HIS is a computerized storage system for patients' health data, including health information, clinical findings, demographics, and billing information [86]. All formal and informal public and private organizations, institutions, and resources that promote, repair, or preserve people's health are included in the healthcare system [87]. Furthermore, a healthcare system includes other stakeholders, such as a grandmother caring for a sick child at home, a private healthcare professional, rehabilitation programs, vector control efforts, health insurance firms, and researchers, to mention a few [88]. Correctly configured HIS assist decision makers in precisely identifying the field's progress, requirements, and challenges. They also enable evidence-based policy and problem-solving decisions. However, HIS in underdeveloped nations, particularly in Sub-Saharan Africa, are inefficient due to a lack of health and IT specialists, rapid population expansion that outpaces available health professionals, high telecommunications costs, civil upheaval and unstable power, and so on [89–91].

4.3. Data Security and Privacy Issues in Healthcare

The healthcare system must balance two competing social benefits to maintain privacy and security. The first is the requirement of proper access and ability to share information for improving care quality, safety, and continuity of treatment. Second is the necessity to establish acceptable methods to protect the privacy of personal healthcare information. However, striking a balance between these two needs is challenging [92]. Despite its benefits, HIS can have issues that prevent them from being used universally in hospitals. One example is the initially high cost of acquiring the fundamental infrastructure required for the HIS. Additionally, in the healthcare industry, privacy and security are still primary considerations, as are the privacy and security of patient data on the computer. Furthermore, medical professionals are notoriously slow to adopt new technology [93].

Privacy, especially in healthcare centers, is a key governing principle of the patient–physician relationship. Patients must share information with their physicians to facilitate correct diagnosis and treatment and avoid adverse drug interactions. Several general security and privacy requirements must be satisfied to provide the appropriate level of privacy for EHR systems; the following have been identified as basic privacy requirements of healthcare centers [94]:

- Access control: The capacity to restrict and control authorized users' access to resources. Identification, authentication, and authorizations are three security and privacy criteria used to determine access.
- Availability: The ability of a system or resource to be accessed and used by authorized users at any time and from any location inside the healthcare system. Preventing service disruptions due to hardware problems, power outages, and system upgrades is also a component of establishing availability.

- Dependability: It ensures that medical data may be easily retrieved at any time, even if hazards are generated by the network dynamics of the failure node. In most medical circumstances, the inability to obtain precise data is caused by network dynamics, which endangers the patient's life. For dependability, fault tolerance is necessary.
- Flexibility: It allows an unauthorized participant not on the permitted list to access specific data in an emergency when needed to save a patient's life. The patient's life may be jeopardized if the access requirements are not followed.

4.4. Security and Privacy-Related Solutions of HIS

Although no two healthcare organizations are the same, many face comparable difficulties regarding data security that result from conflicting desires faced by every modern company: innovation and expansion versus compliance and legal risk mitigation; according to the research, the following five health information system challenges must be resolved [95]:

- The conflict between data expansion and analytics, and data minimization.
- Dealing with mobile applications and related devices.
- Creating adequate cross-functional privacy and security teams.
- The effects of acquisitions on data.
- Effective and tier-based vendor management.

EHRs are replacing paper-based medical records in many hospitals. Higher patient care is now possible due to specialized software and electronic diagnostic equipment. The shift to electronic-based systems allows clinicians to use streamlined, automated processes and customized applications to aid in patient diagnosis and treatment. However, introducing these technologies increases the privacy hazards associated with patient data. When numerous patient records are available electronically, a hostile person attempting to compromise them will be able to readily acquire enormous volumes of data. Some of the most prevalent technological security flaws faced by most healthcare-related firms are as follows [95]:

- Security and privacy policies and plans that are either inadequate or non-existent.
- Continuous security, privacy, and compliance reviews are lacking.
- No backup encryption method is in place.
- The passwords for the operating system, applications, and databases are weak.
- Filtering of material and audit tracking are both lacking.
- Inadequate malware protection exists for viruses, Trojans, spyware, and rootkits.
- Personal health information is disclosed.
- Accountability and duty are lacking.

Some of the current health records management systems are costly, complex, and prone to human error [96]. The researchers proposed the Medicalchain project, which uses the blockchain to manage the health records of registered patients, noting the inability to implement the solution on the SSN system, the Italian National Health Service. A solution envisioned by researchers for healthcare provision in Europe enables sharing of health records, secure integration using blockchain and scalable data lakes, and ensures secure access to data [97]. They have proposed the Serums Smart Health Center System (SHCS) using Hyperledger Fabric and individual data lakes and built a model of access-control model to support collective and individual rules.

Dubovitskaya and others [18] proposed a permissioned blockchain-based system for EHR data sharing and integration and developed the ACTION-EHR system for patient-centric reasons. They presented a framework for the specific data sharing for radiation oncology and granular access control. Mani and others [98] proposed patient-centric healthcare data management (PCHDM) using blockchain. However, they used static access control, so it cannot monitor the healthcare record continuously, which is different from our proposed solution of using a usage control module.

Concerns about patient data privacy and security have led to a low adoption rate of EHRs by several healthcare facilities. Furthermore, EHR research faces difficulties in securing a large amount of sensitive health data in multiple places, and formats [94]. The literature review showed that technical and legal factors, individuals' right to privacy, and policy making are the main challenges facing the development of EHR systems in low- and middle-income countries [99]. However, most of the previous literature only elucidated these problems without reviewing the practical solutions that can be adopted to address them. Despite the importance of BT in improving security and privacy in network systems, including healthcare systems, this field requires more research and development. Moreover, most previous scientific studies and articles were surveys and theoretical studies that focused mainly on the theoretical aspect of healthcare systems and the development of security and privacy in these systems. Very few studies focused on user-centric and privacy solutions. Therefore, the application of BT in healthcare systems to improve data security and privacy is a current topic requiring research and development. This need is what prompted the researchers to conduct this research.

5. Methodology

The authors investigated the literature regarding blockchain and the user-centric privacy of patients' health data records. The author also analyzed and compared the famous blockchain operating systems from several perspectives to choose the optimized one to meet the identified requirements. Based on our analysis, Tables 3 and 4 show the blockchain operating systems' characteristics.

As seen in Tables 3 and 4, we selected various available blockchain operating systems for our analysis, comparing Ethereum, Hyperledger Fabric, ConsenSys Codefi, EOS, Overledger OS, Liberty OS, Ripple, and Corda. We analyzed them regarding language support, blockchain type, consensus algorithm availability, performance, integrated development environment, and transaction flow.

In this analysis, the first aspect we selected is programming language support, as developing this support is the most important of any language. Blockchains use different languages that aid in frontend and smart contract connections [100–104]. Some common programming languages used by blockchains include Solidity, Vyper, Java, Go, Node.js, C++, and Kotlin.

Secondly, the authors analyzed the type of operating system. In our use case, the concern is privacy, so we need to use a private blockchain with complete privacy that a network administrator manages—a permissioned blockchain. Although public blockchains are lighter and faster, they have an open environment, true decentralization, and anonymity. Thirdly, the authors analyzed the consensus algorithm, which achieves reliability in the blockchain network and establishes trust between peers [100,101,103–105].

Fourthly, the authors analyzed the performance [100,101,103–108]. Based on our analysis, EOS has the best performance, followed by Hyperledger Fabric. However, in an EOS implementation, only C++ is available. Although C++ is being developed in the production environment, other scripting languages will provide more availability, support, and implementation in the future. This will limit the future performance of EOS.

Fifthly, the authors analyzed the integrated development environment (IDE), which includes the applications used to develop other applications that include all programming tasks in one application and offer a central interface [100,101,103–105,107,109]. Finally, the authors analyzed the transaction flow or the process from the beginning to the end of the transaction. We found many similarities between the systems in terms of transaction flow.

Since our goal was to develop a complete ecosystem for patient health records, we prioritized the performance, language support, and blockchain type. For this reason, we chose the Hyperledger Fabric, which has neither the highest nor lowest performance, the most accessible language, and high availability of consensus algorithms, including Raft support. It one of the best performing based on a consensus algorithm that is not available in EOS.

Table 3. A comparison of blockchain operating systems and their related features.

Ethereum	Hyperledger Fabric	ConsenSys Codefi	EOS
Programming Language Support			
1. Solidity 2. Vyper	1. JavaScript 2. Java 3. Go 4. Node	1. Solidity 2. Javascript 3. Node	1. C++
Blockchain Type			
Public	Private	Public	Public
Consensus Algorithm			
1. PoW 2. PoS	1. Raft 2. PBFT 3. POET 4. Apache Kafka	1. DeFi	1. DPoS 2. BFT 3. PoW 4. PoS
Performance			
30 transactions per second	3000 transactions per second	30 transactions per second	4000 transactions per second
IDEs			
1. Web:Remix, EthFiddle, ChainIDE, Replit 2. Desktop: Visual Studio Code, Atom, JetBrains IDEs, Hard hat, Brownie, Dapp tools and Truffle	1. Chaincode 2. Visual Studio Code 3. Vagrant 4. Docker containers	1. Remix 2. Visual Studio Code 3. Web2 Stack 4. Web3 Stack 5. Truffle 6. Infura	1. Visual Studio Code 2. EOS Studio 3. EOSIO 4. Swift SDK 5. Java SDK
Transaction Flow			
1. Construct the raw transaction object 2. Sign transaction 3. Validate transaction 4. Broadcast transaction 5. Miner node accepts the transaction, finds block, and broadcasts 7. Local node receives/ syncs the new block	1. Client initiates the transaction. 2. Endorsing peers verify & execute the transaction. 3. Proposal response is inspected. 4. Client assembles endorsements into a transaction. 5. Validate transaction 6. Ledger updated	1. Transaction crafting 2. Gas management 3. Nonce attribution 4. Key signature 5. Transaction sending 6. Transaction listening and event streaming 7. Transaction decoding 8. Transaction response	1. Create transaction 2. Sign transaction 3. Push transaction 4. Verify transaction 5. Authority check 6. Execute transaction 7. Finalize transaction 8. Validate transaction

Table 4. A comparison of blockchain operating systems and their related features.

Overledger OS	Liberty OS	Ripple	Corda
Programming Language Support			
1. Java 2. Javascript	1. Java	1. C++	1. Kotlin
Blockchain Type			
Public	Public	Private	Private
Consensus Algorithm			
1. DLT	1. PoW	1. BFT	1. Quoroboros
Performance			
10 transactions per second	Less than 10 transactions per second	1500 transactions per second	1678 transactions per second
IDEs			
1. SDK Development Kit. 2. Dapp tools	1. WebSphere 2. IBM SDK 3. Eclipse IDE	1. XRP Ledger 2. Visual Studio Code	1. IDE.Corda.net 2. Node explorer 3. VSCode Corda extension 4. Truffle Suite Corda-Flavoured Ganache
Transaction Flow			
1. Prepare a payment transaction 2. Execute a payment transaction 3. Prepare a search for a transaction 4. Execute a search for a transaction 5. Read an overledger transaction 6. Create a subscription for overledger transaction 7. Get a list of subscriptions for Overledger transactions 8. Delete a subscription for an Overledger Transaction	1. Transaction entering 2. Transaction transmitted to the global peer-to-peer computer network 3. Transaction validity verified 4. Confirm legitimate transaction 5. Create a permanent transaction history 6. Transaction completed	1. Submit (send a transaction to the network). 2. Submit_multisigned (Send a multi-signed transaction to the network). 3. Transaction_entry (Retrieve info about a transaction from a particular ledger version). 4. Tx (Retrieve info about a transaction from all the ledgers on hand). 5.tx_history (Retrieve info about all recent transactions)	1. Create proposed transaction 2. Check transaction validity 3. Sign transaction 4. Inspect and verify transaction 5. Commit transaction 6. Record transactions

5.1. Proposed Model

To improve the integrity and privacy of patients' data and allow patients to control their health record access by relevant health personnel, we developed a new framework for healthcare-record systems based on BT and UCON, enabling them to revoke access whenever necessary. Each time the patient allows access to his/her health record, the blockchain component is activated to record the activities. The UCON monitor module checks the permission and provides real-time/continuous access control, giving access to the health record system through blockchain and tracing the records. For this, the UCON model is required as dynamic access control to monitor the runtime access of the patient's health

records and to verify the time constraints placed on the patient's health records. Additionally, the patient can monitor who accessed his/her data and at what time. If an unauthorized access attempt is generated, it will be recorded on an untampered ledger protected by blockchain. Figure 2 shows a diagram of our methodology.

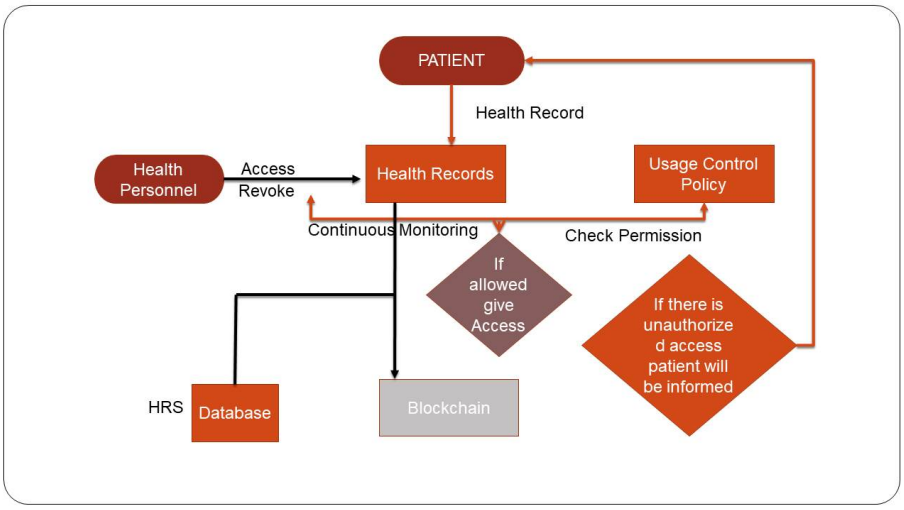

Figure 2. The research methodology explaining our proposal for a user-centric health record system.

5.2. UCON and Patient Health Record Access

The UCON model is required as dynamic access control to monitor the runtime access of the patient's health records. This control will verify the conditions of the time constraints declared by the patient. For example, if the healthcare personnel accesses the health record via a blockchain transaction within a specific time frame, this access will be allowed and constantly monitored at run time. The access will be given for a specific time frame, and the dynamic access will be monitored continuously. As soon as the condition fails, access will be revoked. Similarly, the model grants access to the health record. Additionally, unauthorized access to the health record is recorded, and a notification is sent to keep the record for audit purposes.

Suppose the patient wants to book an appointment at a surgery clinic. In that case, the specified doctor is given access to the patient's health record at a specific time, for example, from 3 to 5 pm, and for a certain number of access times. The patient gives this authority to the doctor. Access control manages the required validity, time, and attempts. It therefore monitors access times, allowing access if the number is not exceeded while revoking access if the number is exceeded or any other unauthorized entry is attempted. These attempts will be monitored and recorded, and notifications will be sent to the patient.

We will have a policy in the HRS that when the doctor needs to access the health record when the patient has an appointment, two hours before the appointment time, the doctor should have access to the patient's health record. If the patient and the doctor agree, the doctor can access the health record. As mentioned in the sequence diagram, we will provide API access such that any HRS can access to the blockchain-based system. Figure 3 shows the sequence diagram for accessing user-centric health records by health personnel.

Similarly, suppose the patient delegates access to a laboratory technician or pharmacist. In that case, access will be granted for a specific time and a specific number of access times, whether to read, write, or both. The attending physician may want to access the patient's health record after conducting analyses to determine specific drugs or the type of operation and ensure that it is performed or postponed. However, the patient will have to grant authority to the doctor again after the specified time. Access control will manage the required authority, time, and access times for each access.

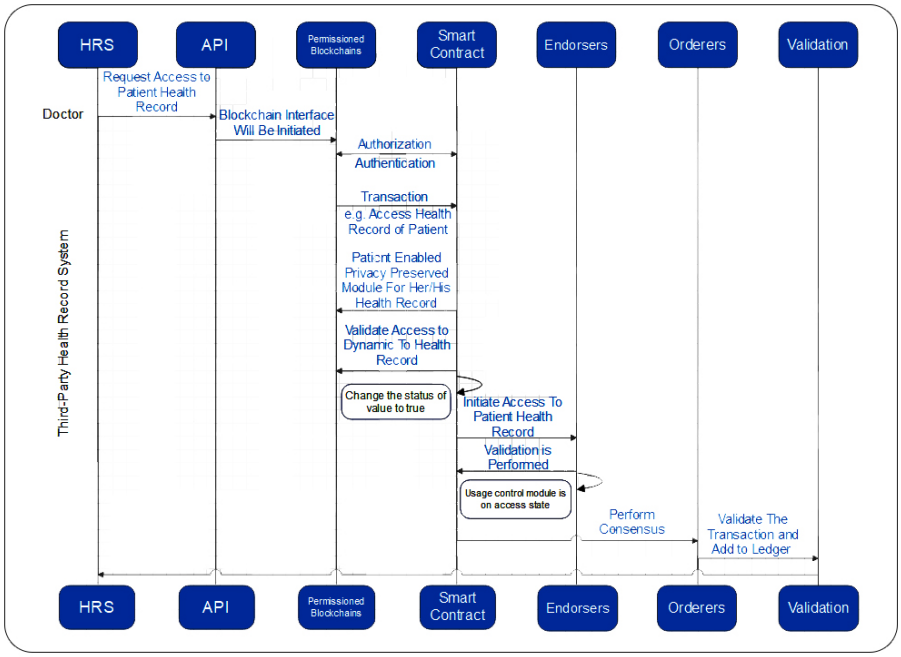

Figure 3. Sequence diagram for accessing the health record of a patient by health personnel, such as a doctor.

If the patient cannot give authority, a family member will be able to grant access authority. Suppose no one is available from the family. In that case, the admin of the blockchain can be given authority to the required doctor at a specific time as an exceptional situation monitored by access control. Notifications will always be sent to the patient in the event of authorized and unauthorized access. Thus, blockchain preserves privacy and monitors all transactions at any time, ensuring they are not changed, tampered with, or deleted.

5.3. Framework Design

We designed the framework for our proposal according to Figure 4.

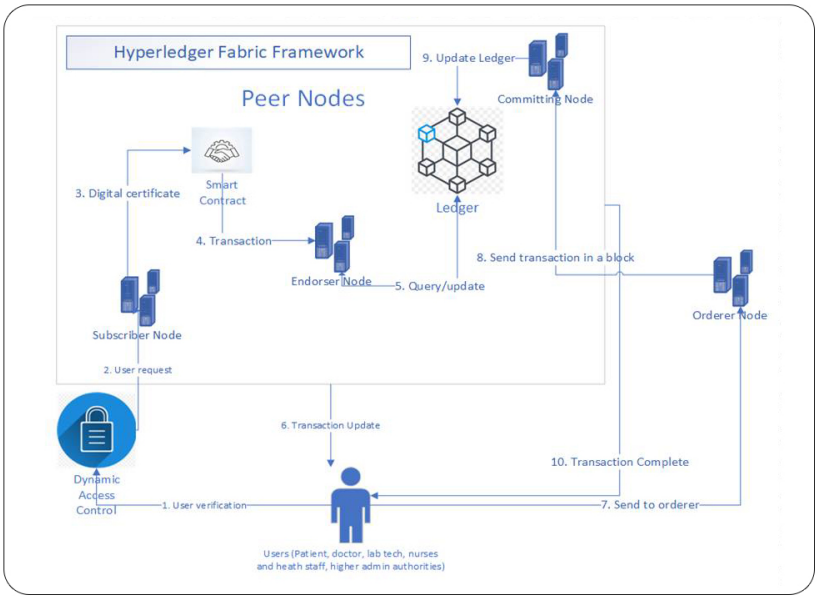

Figure 4. The framework designed using Hyperledger Fabric.

The following illustration describes how Hyperledger Fabric leverages a blockchain to preserve privacy in patient health record access:

- A patient sends a request through a subscriber node, which handles access control by verifying user details.
- If the request is valid, the subscriber node issues a digital certificate and forwards the request to the endorsement node through a smart contract.
- This access control intervention increases security by preventing unauthorized access to the private blockchain.
- Consequently, the endorsement node executes a chaincode to access the ledger and determines whether the transaction is valid.
- The endorsement node signs the proposal and sends its response back to the patient through a smart contract application.
- The transaction is then broadcasted to the ordering node, which creates a block and forwards it to the committing peer (consensus node).
- The committing peer then updates the block to the ledger.
- Invalid transactions are flagged and filtered out during the consensus process, which is handled by the orderer node.
- All information logs are supervised by the patient, who receives feedback responses at the end of each transaction.

5.4. Design of Privacy-Preserving, Patient-Centric Health Record Access Control

We designed a framework using blockchain and other components that are briefly explained in the following definitions. Table 5 summarizes the various symbols and definitions.

Table 5. Definitions and symbols used.

Definition	Symbol
Patient health record	PR
Hospital	HS
Appointment	AP
Doctor	D
Date	DA
Time	T
Blockchain	B
Number of access times	N
Healthcare personnel	HP
Family member	FM
Emergency doctor	ED

5.4.1. Definitions

Definition 1 (create Patient health record)**.** *A patient health record, PR, can be created by the hospital, HS, to ensure the integrity and correctness of the data and the eligibility for treatment.*

Definition 2 (update Patient health record)**.** *A patient health record PR can be updated by the hospital HS to update the data or discover any errors in the registration.*

Definition 3 (choose Patient appointment)**.** *The patient will choose the appointment AP, with the desired doctor D at the specified date, DA, at time T.*

Definition 4 (set access policy)**.** *The patient will permit the specified doctor D to access the health record during the pre-set time T and for a specified number of access times N.*

Definition 5 (Blockchain access)**.** *The blockchain admin B will store the health record in the clinic of doctor D in time T after the permission of patient P.*

Definition 6 (Delegate access)**.** *The doctor D can delegate access to other healthcare personnel HP if tests or other requests are needed.*

Definition 7 (Emergency case)**.** *A family member FM will permit emergency doctor ED in time T for a specified number of access N if the patient is in an emergency or has an accident. If all family members suffer from the same accident, permission will be given directly to the emergency doctor ED by the blockchain admin B at the specified time T to save the patient's life.*

5.4.2. Algorithm

The flow of information and details of how the patient-centric health records access control framework uses blockchain through a generic algorithm is shown below in Algorithm 1.

Algorithm 1 Patient-centric health record access control in a blockchain.

The hospital HS creates or updates a patient health record PR
patient P chooses an appointment with doctor D during time T on date DA
Patient P gives permission to doctor D
if *Permission == AllOW AND Role == DOCTOR* **then**
    Access is given with number of access N in time T "read and write access"
else
    Permission=Deny
end if
Blockchain Admin B takes the permission from patient P and stores the health record in the clinic of doctor D in time T
Doctor performs lab tests for patient P during same time T
Delegate permission to HEALTHCARE PERSONNEL HP "such as lab technician or pharmacist"
if *Permission == AllOW AND Role == HEALTHPERSON* **then**
    Give access with number of access N in time T "read-only access"
else
    Permission=Deny
end if
HEALTHCARE PERSONNEL HP needs to update the lab results in the patient's health record "write access"
if *Permission == AllOW AND Role == HEALTHPERSON* **then**
    Give access with number of access N in time T "read and write access"
else
    Permission=Deny
end if
Doctor D needs to change the medicine after the result is finished and time T is finished
Patient P Gives permission to doctor D
if *Permission == AllOW AND Role == DOCTOR* **then**
    Give access with number of access N in time T "read and write access"
else
    Permission=Deny
end if
Blockchain Admin B takes the permission from the patient P and stores the health record in the clinic of doctor D in time T
Family member FM Gives permission to emergency doctor ED when patient P is in an emergency situation or has had an accident
if *Permission == AllOW AND Role == EMERGENCYDOCTOR* **then**
    Give access with number of access N in time T "read and write access"
else
    Permission=Deny
end if
Blockchain Admin B takes the permission from patient P and stores the health record in the Emergency area of emergency doctor ED in time T
Blockchain admin B Gives access with Number of access N in time T when Family members FM are suffer to the same accident "read and write access"
if *Permission == AllOW AND Role == EMERGENCYDOCTOR* **then**
    Give access with number of access N in time T "read and write access"
else
    Permission=Deny
end if

6. Implementation

We chose Hyperledger Fabric to start our design, as mentioned in Section 5, due to its performance, blockchain type, integrated development environment, transaction flow, and language support; the variety of modular design of the Hyperledger operating system; and ease of smart contract development with it.

Our framework was built with blockchain network entities and a smart contract called (chaincode) in Hyperledger Fabric. Chaincode is the application running in the form of a smart contract, and various transactions can be executed throw that over the blockchain network.

6.1. Testbed Environment

The Implementations were carried out on ORACLE VM (VirtualBox Graphical User Interface version 6.1.26 r 145957 (Qt50602)) with Ubuntu virtual machine version 20.4.4 (Linux), 64-bit operating system. Table 6 shows the other configurations.

Table 6. Testbed environment.

RAM	3005 MB
Processor	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz 2.00 GHz
Network Card	Intel corporation 82540EM Gigabit Ethernet controller (rev 02)
Disk Capacity	85.9 GB
Architecture	x86-64
CPU(s)	2
Core(s)	4
Hyperledger Fabric	v2.3.0

6.1.1. Smart Contract Installation

The first step was to install the smart contract we have created for patients, doctors, lab techs, other health personnel, and administration. As can be seen in Figure 5, Fabric release 2.3.0 was installed. After that, we needed the peer nodes and channels. As we explained in the background, the channel is a necessary component of any blockchain network, which is created as the first entity to allow communication between members, and it contains a chaincode, nodes, and clients or members created with a set of credentials assigned via (membership service).

Once the channel is ready and installed, our peer nodes are initiated. The very first peer is a database. We have many types of databases supported by Hyperledger, such as LevelDB and CouchDB. However, we configured CouchDB for our prototype implementation, a NoSQL database that allows open-source queries to collect and store data in JSON format. Additionally, we have many languages supported by Hyperledger for the chaincode, such as Java, Go, and Node.js. We used Node.js, the most widely used language, making it easy to interact with APIs with Hyperledger Fabric for our proposed patient-centric smart contract.

We configured our chaincode name as patientcc with version 1.0, so we could initiate, approve, commit, initialize, and discover the chaincode for patients-related healthcare records, doctors, other health personnel, and administration staff. Block number is . Later, we will explain how to configure this block number with multiple sizes, and we will explain that in the results and what the performance affects. To begin our development and implementation, we installed our smart contract (patientcc). Figures 5–7 show installation information.

```
user1@user1:~/HLF$ ./minifab up -l node -s couchdb -e true -n patientcc -d false -p false
Using default spec file
Minifab Execution Context:
   FABRIC_RELEASE=2.3.0
   CHANNEL_NAME=mychannel
   PEER_DATABASE_TYPE=couchdb
   CHAINCODE_LANGUAGE=node
   CHAINCODE_NAME=patientcc
   CHAINCODE_VERSION=1.0
   CHAINCODE_INIT_REQUIRED=false
   CHAINCODE_PARAMETERS=false
   CHAINCODE_PRIVATE=false
   CHAINCODE_POLICY=
   TRANSIENT_DATA=
   BLOCK_NUMBER=newest
   EXPOSE_ENDPOINTS=true
   CURRENT_ORG=org0.example.com
   HOST_ADDRESSES=192.168.1.113
   WORKING_DIRECTORY: /home/user1/HLF
```

Figure 5. Smart contract installation information.

```
.......
# Preparing for the following operations: ********************
  verify options, download images, generate certificates, start network, networ
 status, channel create, channel join, anchor update, profile generation, cc in
tall, cc approve, cc commit, cc initialize, discover
...................
# Running operation: ****************************************
  verify options
.
# Running operation: ****************************************
  download images
............
# Running operation: ****************************************
  generate certificates
.....................................................................................
.............................................................
# Running operation: ****************************************
  start network
.........................
# Running operation: ****************************************
  network status
```

Figure 6. Smart contract operation information.

```
# Fabric network peer and orderer node health status **********
  peer1.org0.example.com "OK"
  peer2.org0.example.com "OK"
  peer1.org1.example.com "OK"
  peer2.org1.example.com "OK"
  orderer1.example.com "OK"
  orderer2.example.com "OK"
  orderer3.example.com "OK"
  Network Status: 100%
```

Figure 7. Smart contract network information.

Figures 5 and 7 display the following important information, which will be shown in Table 7.

Table 7. Installation information.

Fabric release used	2.3.0
Database type	couchdb
Channel name	mychannel
Language	Node.js
Smart contract name	patientcc
Network organisation	0.example.com
The solution included two organisations	(org0 and org1)
The solution included	four peers and three orderers

The solution also included a Hyperledger explorer—a Web application tool used to view, invoke, and query blocks, transactions and related data, network information, and chaincode. Additionally, Hyperledger explorer provides dashboards with tables and graphs containing information on our network, blocks, transaction, chaincode, performance, channel information, and organization, which help make decisions. Additionally, we have the numbers of blocks and transactions represented in blocks/hour, blocks/min, tx/hour, and tx/min. For our performance reason, we initiated these transactions, and will explain that in the result and what the performance affects. Figures 8 and 9 display the dashboard information.

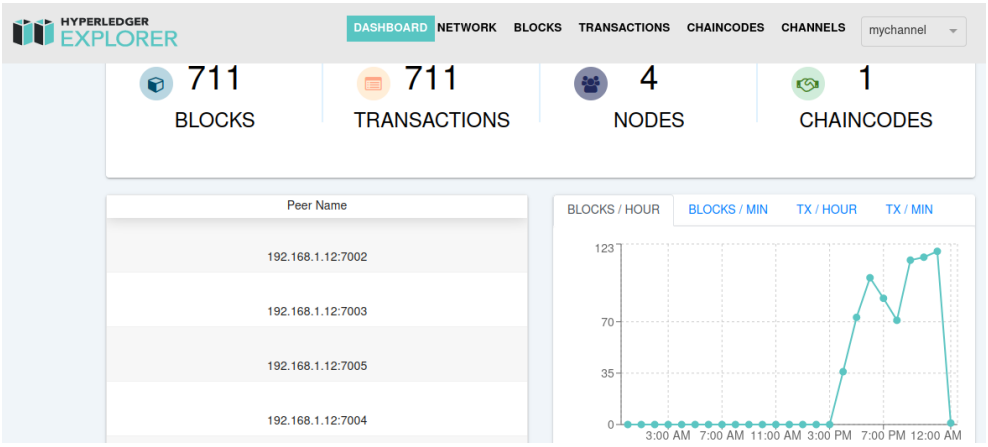

Figure 8. Dashboard 1 smart-contract information.

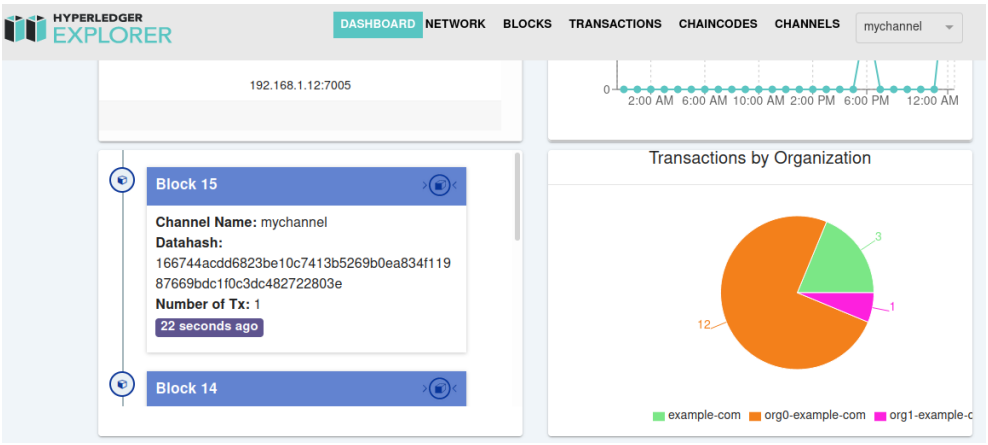

Figure 9. Dashboard 2 smart-contract information.

6.1.2. Hyperledger Components

We have the stakeholders in the first layer of Hyperledger Fabric. Specific rules apply to the smart contract for the different stakeholders, represented by patients who own their health records; doctors; healthcare personnel such as lab techs, pharmacists, and nurses; administration staff; the API; and any third party that needs to connect with our network. In the second layer, we have the smart contract, which initiates transactions between clients or participants through the network. Those transactions must be reliable, traceable, and immutable or tamper-proof.

Hyperledger Fabric comprises peers (endorsers, orderers, consensus, and anchors).

There are plenty of HRS-based applications configured in various hospitals and clinics around the globe. It would be difficult to change the existing infrastructure and adopt our proposed solution in all the HRS in the world. We have given the flexibility to those patients who would like to use our user-centric proposed solution, so we have provided third-party API integration with the existing HRS. Hence, the patients can be more relaxed about using

the user-centric application without changing the existing HRS system, and they will be able to interact with our proposed system. Additionally, users such as doctors can use it without creating other accounts.

The anchor peer (orderer node) is essential because every peer node in the network contacts it to confirm the presence and get the updates in the network. The endorsement policy assigns that a desired number of peers must sign every transaction before being listed in the ledger, which means the transaction is validated by the peers. We defined multiple endorsement policies such as transactions coming from a patient for a new appointment, the doctor to access the health record, healthcare personnel to access or update the health record, and a family member to give access in emergency cases. Additionally, the endorsement policy invokes the validated system chaincode (VSCC), which is automatically initiated when the transaction reaches the peer and checks for the transaction source and if the transaction is signed by the desired number of entities and the correct number of endorsements.

The certification authority (CA) is used to verify the owners in the blockchain network, make secure network communication by generating the SSL certificate and private and public keys for all peers, and provide access to add peers. All client messages are ciphered using cryptographic keys to provide reliability. Additionally, all data transferred from a client or user are digitally signed by the user's private key, which is encrypted by the public key of the receiver user, which means the confidentiality and integrity of data are preserved with privacy.

6.1.3. Patient's Perspective (Stakeholder)

When users or clients in a blockchain network perform an action in the form of a transaction, such as taking an appointment, a transaction is initiated. That is verified via the endorsers and goes to the orderer node. Once the orderer node receives the number of the transaction, a block is created according to the configured size of the block. Finally, the block's hash is generated and stored on the ledger via the consensus protocol. Thus, in the case of RAFT, the block is given to the leader node of the network. The leader synchronizes the hash in the ledger.

Initially, the patient who generated the transaction is validated (by ensuring that he/she has an active account). The patient's selected slot is also verified (whether the specific time is available or not). Once the patient is approved for the appointment, he/she can mention the health record to healthcare personnel.

6.1.4. Healthcare Personnel Perspective (Stakeholder)

After the patient books an appointment with a specific doctor, he/she will have to choose the number of times the same doctor accesses his health record during the preselected period. The doctor will then be able to access the health record. This will be done by creating a transaction in the form of a set access policy to the doctor. That is verified via the endorsers and goes to the orderer node. Thus, the block is created, and the block's hash is generated and stored on the ledger via the consensus protocol.

When the doctor accesses the policy, that will be validated through our UCON model. As mentioned in the background, the UCON model describes a user's session in three phases: pre-, ongoing, and post-access. Once the doctor receives access via a blockchain transaction within a specific time frame, it will be in the access stage. It will be allowed, and the UCON monitor module checks the permission and provides real-time/continuous access control. The access will be revoked when the time window ends. Traditionally, access-control models are static in the blockchain network, so once permission has been granted, there is no way to revoke access. Thus, this type of access is not available in a traditional blockchain system, and we have created a small means to implement UCON, which is dynamic access control. The patient will be able to give specific access to his health record in this case.

Patients can also grant access to healthcare personnel, such as lab technicians or pharmacists, in case a doctor needs to delegate access to other healthcare personnel if tests or other requests are needed. After patient approval, the healthcare personnel can write and read patient records. Each time, a transaction is created and verified. All transactions are given to the endorsers in the blockchain network, which validate each in the system. After patient approval, the healthcare personnel can write and read patient records.

One of the additional features of our proposed solution is that in addition to the privacy of the patient, if anything is miscommunicated between healthcare personnel—or, for example, if medications are misprescribed by the doctor or there is a side effect of the medication the patient did not know about—that will be a part of the record in the blockchain. It will not be modified due to these blockchain characteristics so that everything will be transparent. Thus, any dispute between the patient and doctor, between the patient and healthcare personnel, or between the doctor and healthcare personnel, could be easily identifiable and could not be denied by anyone.

Figure 10 shows the smart contract and transaction of our proposed solution.

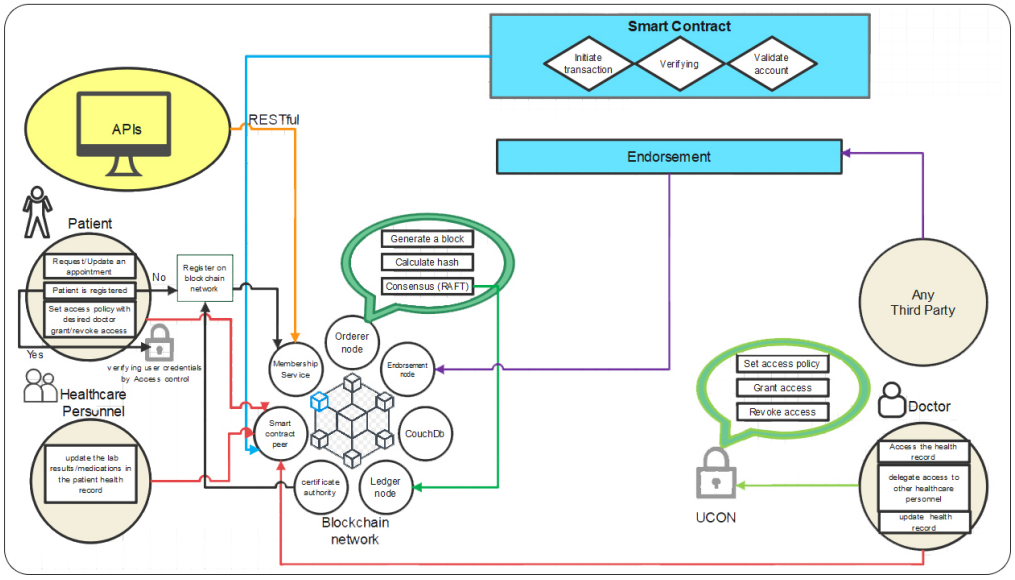

Figure 10. Smart contract and transaction over Hyperledger Fabric .

6.1.5. Smart Contract Code

Regarding our goal of designing a framework for healthcare-record systems based on BT and UCON to improve the integrity and privacy of patients' data, allow patients to control their health record access by relevant health personnel, and to implement user-centric privacy, preserved by a secure HRS system on top of the blockchain framework, we designed the necessary code for implementing the proposed design.

We used the Node.js language to design the smart contract, insert the access control, and explain the working mechanism; see the code shown in Figures 11–16. The code of the smart contract can be accessed on this link to GitHub: https://github.com/RayanAbutaleb/chaincode-patientcc.git, (accessed on 3 January 2023).

6.1.6. Registering Users

All the stakeholders should be registered as members in the HRS, as Hyperledger provides a RESTful interface, so stakeholders will log in with the same interface, and we have plugged in the username and password link with (API), so it can be justified through blockchain.

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Contract } = require('fabric-contract-api');

class PatientrecordContract extends Contract {

    async policyExists(ctx, policyId) {
        const buffer = await ctx.stub.getState(policyId);
        return (!!buffer && buffer.length > 0);
    }

    async setAccessPolicy(ctx, patientId, organizationId, readconstraint) {
        const policyId = patientId+organizationId+'policy';
        const exists = await this.policyExists(ctx, policyId);
        if (exists) {
            throw new Error(`Access Policy already defined for patientId: ${patientId}`);
        }
        const constraints = { readconstraint };
        const buffer = Buffer.from(JSON.stringify(constraints));
        await ctx.stub.putState(policyId, buffer);
        const clientOrg = ctx.clientIdentity.getMSPID();
        console.info('============= Access Policy has been set for ===========', clientOrg,
'on patient record id: ', patientId);
    }
```

Figure 11. Smart-contract code using Node.js.

```

    async readAccessPolicy(ctx, patientId, organizationId) {
        const policyId = patientId+organizationId+'policy';
        const exists = await this.policyExists(ctx, policyId);
        if (!exists) {
            throw new Error(`No Policy define for patientId: ${patientId} does not exist`);
        }
        const buffer = await ctx.stub.getState(policyId);
        const asset = JSON.parse(buffer.toString());
        const clientOrg = ctx.clientIdentity.getMSPID();
        console.info('============= Read Acccess Policy ===========', policyId);
        return asset;
    }

    async createHistory(ctx, patientId, organizationId, value='read') {
        const historyId = patientId+organizationId+value+'history';
        const asset = { value };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(historyId, buffer);
        console.info('============= History is created ===========', historyId);
    }

    async patientrecordExists(ctx, patientrecordId) {
        const buffer = await ctx.stub.getState(patientrecordId);
        return (!!buffer && buffer.length > 0);
    }

```

Figure 12. Smart-contract code using Node.js.

```
    async createPatientrecord(ctx, patientrecordId, value) {
        const exists = await this.patientrecordExists(ctx, patientrecordId);
        if (exists) {
            throw new Error(`The patientrecord ${patientrecordId} already exists`);
        }
        const asset = { value };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(patientrecordId, buffer);
    }

    async readPatientrecord(ctx, patientrecordId) {
        const exists = await this.patientrecordExists(ctx, patientrecordId);
        if (!exists) {
            throw new Error(`The patientrecord ${patientrecordId} does not exist`);
        }
        //------------------------------------------------
        const clientOrg = ctx.clientIdentity.getMSPID();
        const policyId = patientrecordId+clientOrg+'policy';
        //------------------------------------------------------
        var policy_limit = 0;
        const buffer1 = await ctx.stub.getState(policyId);
        if (buffer1.length > 0){
                const policy = JSON.parse(buffer1.toString());
                const final_policy = JSON.stringify(policy);
                const obj = JSON.parse(final_policy);
                policy_limit = obj.readconstraint;
                console.info('=======POLICY========', final_policy, '--', policy_limit);
```

Figure 13. Smart-contract code using Node.js.

```
        }
        //---------------------------------------------------
        const historyId = patientrecordId+clientOrg+'readhistory';
        let iterator = await ctx.stub.getHistoryForKey(historyId);

        let result = [];
        let res = await iterator.next();
        while (!res.done) {
                if (res.value) {
                        console.info(`found state update with value: $
{res.value.value.toString('utf8')}`);
                        const obj = JSON.parse(res.value.value.toString('utf8'));
                        result.push(obj);
                }
                res = await iterator.next();
        }
        await iterator.close();
        //---------------------------------------------------
        const no_read_request = result.length;
        console.info('=======History11========', historyId, ' ', no_read_request,
policy_limit);
        if (no_read_request > policy_limit){
                throw new Error(`${clientOrg} exceeded read-access limit ${policy_limit} for
accessing patientrecord: ${patientrecordId}`);
        }
        //---------------------------------------------------
        await this.createHistory(ctx, patientrecordId, clientOrg, 'read');
        //---------------------------------------------------
```

Figure 14. Smart-contract code using Node.js.

```
        const buffer = await ctx.stub.getState(patientrecordId);
        const asset = JSON.parse(buffer.toString());
        return asset;
    }


    async updatePatientrecord(ctx, patientrecordId, newValue) {
        const exists = await this.patientrecordExists(ctx, patientrecordId);
        if (!exists) {
            throw new Error(`The patientrecord ${patientrecordId} does not exist`);
        }
        const asset = { value: newValue };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(patientrecordId, buffer);
        //---------------------------------------------------
        const clientOrg = ctx.clientIdentity.getMSPID();
        await this.createHistory(ctx, patientrecordId, clientOrg, 'update');
        //---------------------------------------------------
    }

    async deletePatientrecord(ctx, patientrecordId) {
        const exists = await this.patientrecordExists(ctx, patientrecordId);
        if (!exists) {
            throw new Error(`The patientrecord ${patientrecordId} does not exist`);
        }
        await ctx.stub.deleteState(patientrecordId);
        //---------------------------------------------------
```

Figure 15. Smart-contract code using Node.js.

```
        const clientOrg = ctx.clientIdentity.getMSPID();
        await this.createHistory(ctx, patientrecordId, clientOrg, 'delete');
        //---------------------------------------------------

    }
}

module.exports = PatientrecordContract;
```

Figure 16. Smart-contract code using Node.js.

6.1.7. Creating Patient Records

At first, the hospital can create the patients' health records, and all users will be registered in the network and have unique IDs. All of this will be managed through the blockchain. We ran the transaction (create patient record) with the required argument.

6.1.8. Retrieving Patient Records

In the event of the need to retrieve the data of any patient to use his data for any purpose, we could retrieve the patient's name using his/her value or ID.

6.1.9. Update and Delete Patient Records

The hospital or blockchain admin can update the patient's record with the correct name or ID, or any wrong record can be disabled but not deleted due to the blockchain's

characteristics, achieving security, openness and transparency, de-trusting, cryptography, anonymity, audibility, and a tamper-proof state.

6.1.10. Creating an Access Policy

The patient has full control to give permissions such as read, write, and deny access to users on the health record. After the patient chooses the appointment, access control is applied, and the patient will grant the necessary permissions to the relevant doctors or health personnel a specified number of times. This permission in a amount of time from the patient expresses a user-centric nature.

7. Results

Blockchains transform entire industries through the promises of decentralization and immutability, which enhance data integrity and security. The novel technology uses consensus mechanisms to support multiple use cases in finance, banking, regulatory compliance and auditing, capital markets, healthcare, and other fields. Despite the efforts vested in developing this technology, achieving optimal performance, privacy, and scalability are still challenging in different deployment environments [110]. These aspects of the blockchain should be fully advanced to enhance enterprise application feasibility. In contrast to Bitcoin and Ethereum, which allow public access, most enterprises utilize permissioned distributed ledger technologies (DLTs) to limit access to transactions to an organization or conglomerate. DLTs became popular with the advent of Hyperledger and R3 projects in 2015, though the need to boost their performance to outmatch centralized systems persists.

This paper evaluates the performance of Hyperledger Fabric from research and healthcare sector project development standpoints. It includes a range of performance parameters used in other studies and custom scenarios relevant to the health sector, such as integrity, privacy, availability, broad-scale deployments, a cross-data center, and resilience. The in-depth analysis aims to reaffirm enterprise blockchains as highly-complex and fault-tolerant distributed frameworks that are effective in real-life scenarios.

7.1. Distributed Ledger Technology

Traditionally, manual and digital ledgers have been utilized to record business transactions and have been preferred for immutability. The blockchain extends the ledger concept from a centralized to a decentralized system, eliminating users' need to trust a single entity. The distributed system allows multiple entities to determine the ledger's progression through a consensus approach, such as PoW, which is used in Bitcoin. Unlike open systems, permissioned ledgers do not require a strict trust model because users know each other and access is restricted. Since specific breaches such as spamming the ledger are less anticipated, the consensus mechanisms in permissioned environments may significantly differ from those of permissioned ledgers. The participants in a restricted environment need an identical copy of the ledger to achieve to concede.

Besides transactions, DLTs also support smart contracts, which perform complex operations and require conditional execution of transactions. Smart contracts represent legal contracts by defining in-built rules through code. While DLTs permit secure transactions, scalability and performance challenges inhibit their uptake. Therefore, designing DLT-based frameworks requires quantifying key metrics, such as latency and throughput. This analysis utilized the Distributed Ledger Performance Scan (DLPS) for transparent and automatic performance evaluation. The benchmarking solution improves scalability and performance and encourages the adoption of DLTs.

7.2. Hyperledger Fabric

Hyperledger Fabric is a framework for implementing a permissioned distributed ledger with an extensible and modular architecture. It facilitates exchanging consensus protocol and authentication mechanisms and permits customization for multiple use cases. It is an open-source project by the Linux Foundation and is therefore improved quickly

and regularly by a large community of developers. In Fabric, smart contracts, also called chaincodes, are executed to modify the ledger's state. When the chain code is deployed and executed, the code supports two public functions: $Int()$ and $Invoke()$.

7.3. Experiment Environment

Different platforms help evaluate DLTs. Two of them are simulation and emulation. Simulators use virtual systems to mimic the interactions of various components to provide a reproducible and controlled environment for experimentation. The network is scalable because no hardware or real network is involved. Moreover, it is possible to confine the evaluation to a single machine, as was the case in this analysis, to facilitate debugging, time manipulation, and workload control. Emulators are hybrid platforms with both real and virtual network components. Their output is real and accurate but less reproducible, and the platforms require additional overhead.

In the prototype, we have as candidates the patient, doctors, lab techs, nurses and health staff, and administrative users. The implementation was carried out on ORACLE VM (VirtualBox Graphical User Interface, version 6.1.26 r 145957 (Qt50602)) with an Ubuntu virtual machine, version 20.4.4 (Linux), on a 64-bit operating system. The virtual machine was deployed on a single computer for simulation. The hardware specifications included 3005 MB RAM, an Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz 2.00 GHz processor, an Intel Corporation 82540EM Gigabit Ethernet controller (rev 02) network card, and a hard disk capacity of 85.9 GB. Additionally, two CPUs and four cores of x86-64 architecture were used. The evaluation framework included Hyperledger Fabric v2.3.0, a CouchDB database, a channel (named mychannel), a network organization (0.example.com), two organizations (org0 and org1), four peers, and three orderers.

To measure the impacts on the performance metrics, we considered five incoming transaction rates (create a patient record, read a patient record, update a patient record, delete a patient record, and set access policy), ranging from one transaction per second (tps) to 1000 tps. Figure 17 shows the process flow of the patient record scenario. Figure 18 shows the number of transactions.

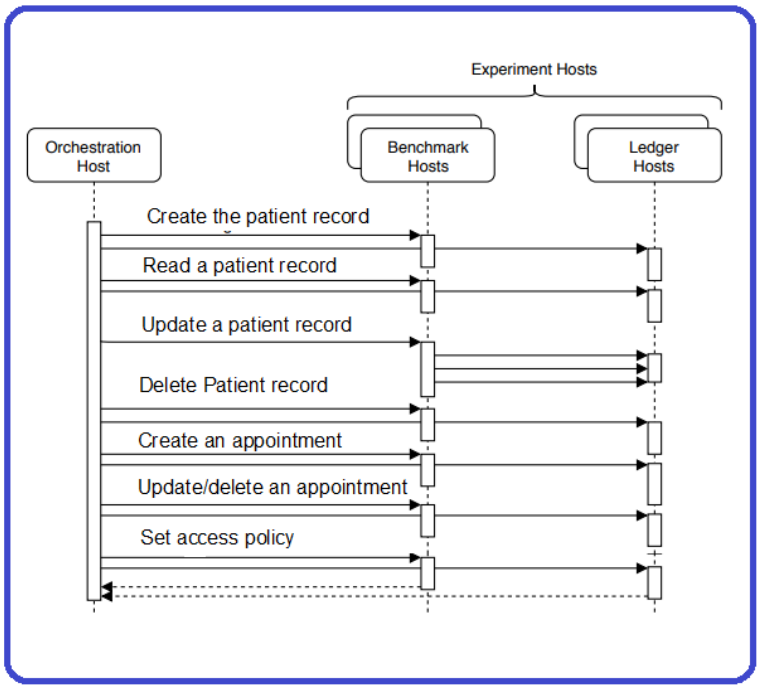

Figure 17. Process flow for patient record scenario.

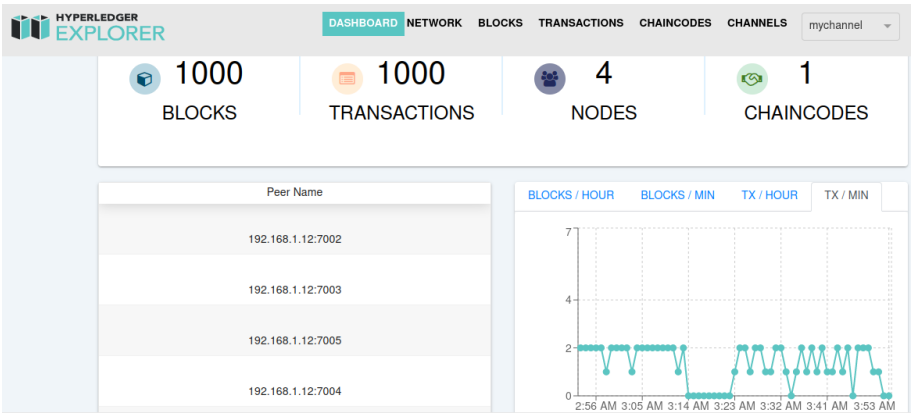

Figure 18. Number of transactions.

7.4. Related Work

The performance of blockchains is fundamental to enterprise-based systems that require high throughput and low latency. Therefore, various researchers have examined the efficiency of blockchains in different dimensions. Table 8 is a summary of related work [111–117].

Table 8. Summary of the performance evaluation of Hyperledger fabric.

Source	Detailed Content
Nasir et al.	This article is a comparative performance analysis of two versions of Hyperledger Fabric: v.06 and v.1.0. The analysis focused on various metrics, including scalability, security, execution time, throughput, and latency. The findings show that Hyperledger Fabric v.1.0 outperforms v.0.6 consistently in all aspects though it never attained the performance levels registered in traditional databases under high workloads.
Baliga et al.	In this research paper, the authors characterize the performance of Hyperledger Fabric v.1.0 under various workloads. The study involved experiments examining the latency and throughput using a set of micro-benchmarks with varying chaincode and transaction parameters. Additionally, the researchers evaluate the platform's performance under an increasing number of chaincodes, peers, and channels.
Thakkar et al.	The article examines the performance of Hyperledger Fabric and identifies bottlenecks in two phases. The initial phase involves using various setup configurations, such as endorsement policy, block type, CPU allocation, state database, multi-channels, and latency for assessment. The bottlenecks identified include validation and verification of sequence block, endorsement policy, and state database. The authors enhanced the overall throughput 16 times by optimizing existing bulk read/write operations for CouchDB.
Javaid et al.	In this paper, the researchers demonstrate the performance improvements in CouchDB and LevelDB after restructuring the validation phase of Hyperledger Fabric. The modified Fabric utilizes a chaincode cache, reads the state database in parallel with validated transactions, and updates the database and ledger in parallel. The results indicate 2× and 1.3× improvements in CouchDB and LevelDB, respectively.
Sharma et al.	The researchers examined a health system that preserves the privacy of electronic health records generated from Internet Of Things (IoT) devices. The system, referred to as kHealth, uses differential privacy and homomorphic encryption to secure records. The authors note that optimizing scalability increases susceptibility to privacy breaches.
Dubovitskaya et al.	The article presents a scalable, privacy-aware, cloud-based electronic health system that uses private key cryptography on local and cloud databases. The medical data is efficiently decentralized through patient-specified access control. The system may face privacy challenges due to untrustworthy cloud-based server providers that can deduce personal information from the user's IP addresses, thereby breaching anonymity.
Roehrs et al.	Authors used the openPHR protocol to distribute electronic health records into data blocks among participating devices. The protocol was proven elastic, feasible, and generalizable. to many organizations. Although their architecture is described in detail, the researchers admit a lack of privacy and security in their technique.

7.5. Evaluation Framework

7.5.1. Security Evaluation

Hyperledger Fabric, like other technologies that use certificates and identities, can easily be compromised by users. Disclosure or theft of a blockchain user's private certificate could enable a malicious user to perform read and write instructions on the ledger. Depending on the existing policy, system security may be at risk. In this project, such a scenario is unrealistic because ordinary users and administrators lack full control over the ledger. The chaincode that runs autonomously in all the peers may also pose a system risk due to bugs. Therefore, the system should be inspected and tested for bugs. Additionally, quantum computing seriously threatens systems that use traditional encryption techniques and hashing methods. Hence, robust countermeasures are necessary to address such threats.

7.5.2. Anonymity Evaluation

A privacy-aware framework ensures anonymity and unlinkability between clients and transactions. Evaluating anonymity involves retrieving crucial information during user interaction and chaincode execution. The client identity library built into the chaincode aids in analyzing disclosed identity attributes in various interactions. The library provides a way to access private validator information using the transport-layer security protocol for encryption. Blockchains are typically assigned public keys that conceal confidential data.

Since Hyperledger Fabric offers a large configuration space, various parameters should be configured for anonymity. Moreover, a blockchain network's performance should be measured by various static variables, including the endorsement policy and the numbers of orderers, peers, and organizations. At least one organizational peer should sign each transaction proposal to prevent unnecessary interactions between endorsing parties. Two organizations (org1 and org2) were established with different technical constraints and levels of network complexity. The organization comprised three validating peers and four orderers.

7.5.3. Performance Evaluation

The DLPS was used for standardized benchmarking and to analyze influential variables and the performance of Fabric. The tool was chosen because it clearly defines how it determined throughput and latency, which are the key performance metrics in this study. The DLPS is open-source and facilitates the testing of various configurations through its cloud network support. The benchmarking covered several variables that potentially influence the performance of Hyperledger Fabric. Since DLPS did not support all Fabric properties by default, the latter was upgraded to Fabric 2.0 to support multiple channels, private transactions, and complex queries. Moreover, the CouchDB and ordering node docker were set on separate or the same peer nodes to increase the support for architectural parameters. Splitting or joining tasks with different peers potentially enhances performance by lowering cross-instance latencies.

Additionally, the design supported multiple data center deployments and network delay simulation. The benchmarking processes initiated automatic crashes of orders and peers and assessed traffic statistics and single-core CPU usage. The variables considered for benchmarking are summarized in Table 9.

The experimental testing was incremental to enhance the reliability of outcomes. A configuration file specified the particularities of the Hyperledger Fabric, and the DLPS utilized the file to set the client network and blockchain before the benchmarking processes. In a single DLPS run, requests were sent from clients to the network for a specific duration and rate, which is portrayed as the gradient of requests on a curve. A response curve was generated to confirm successful transaction processing. Seven incoming transaction rates were considered, including creating the patient record, reading a patient record, updating a patient record, deleting a patient record, creating an appointment, updating or deleting an appointment, and setting an access policy. Figure 19 illustrates the single benchmarking run, and Figure 20 portrays the ramp series.

Table 9. Variables.

Group	Variable
Architecture	Endorsement policy Database Location Number of orderers, peers, and organizations Number of channels
Setup	Block parameters Hardware Database type
Business Logic	I/O-heavy workload Private data Reading versus writing CPU-heavy workload
Network	Bandwidth Delays

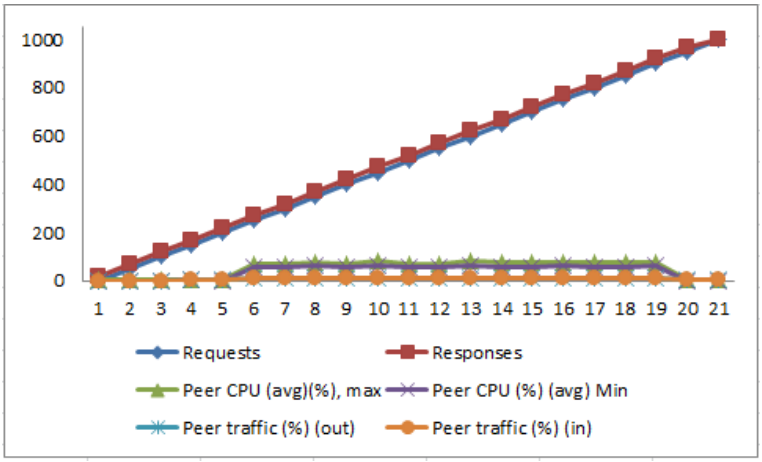

Figure 19. Single benchmarking run.

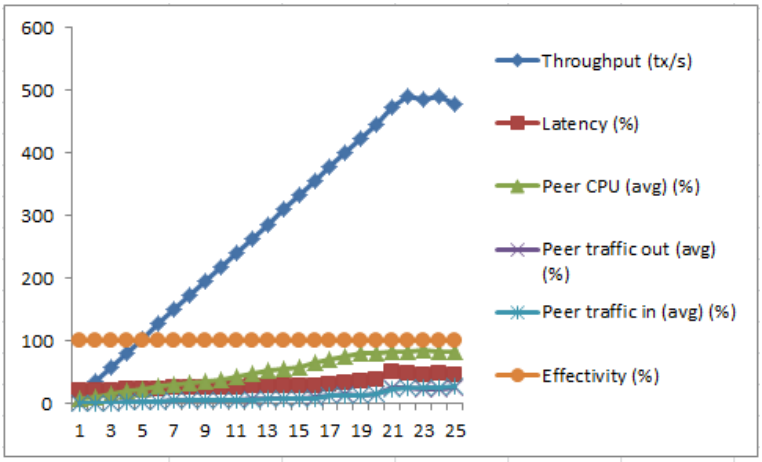

Figure 20. Benchmarking ramp series.

7.6. Benchmarking Result

Default Setup

Considering the modifications to the default architecture regarding the parameters under study and the outcomes from reviewed literature, it was clear that changing the

numbers of clients, channels, and ordering services did not affect throughput. Guggenberger et al. [118] found that doubling the number of clients to spread the workload improved the performance of private transactions by 4% but did not affect the performance of public transactions (a public transaction means anyone with access to a blockchain network can examine the details of the transaction; in contrast, a private transaction means only specific participants can examine the details of the transaction). Increasing the number of channels by the same magnitude increased private performance and lowered public performance, by 13% and 2%, respectively, (14). Therefore, relying on a single channel in this project did not undermine the performance outcome in terms of throughput. CPU utilization among peers on all cores was at maximum with the single channel, implying that an additional channel would not achieve higher throughput.

7.7. Architecture

7.7.1. Endorsement Policy

The endorsement policy is crucial because it influences the level of redundancy. Increasing endorsers improves robustness at a higher overhead cost. A rise in endorsers lowers throughput. Increasing orderers lowers the performance of CouchDB by 14% for simple public transactions and by 41% if the number of endorsers is doubled (Figure 21). For private organizations, the private transactions between org1 and org2 were evaluated accordingly. Doubling the number of endorsers from two to four resulted in a 14% loss in CouchDB throughput. Public transactions yielded a 31% decrease in throughput for the same database. Therefore, increasing the number of endorsers in CouchDB affected public transactions more grossly than private transactions.

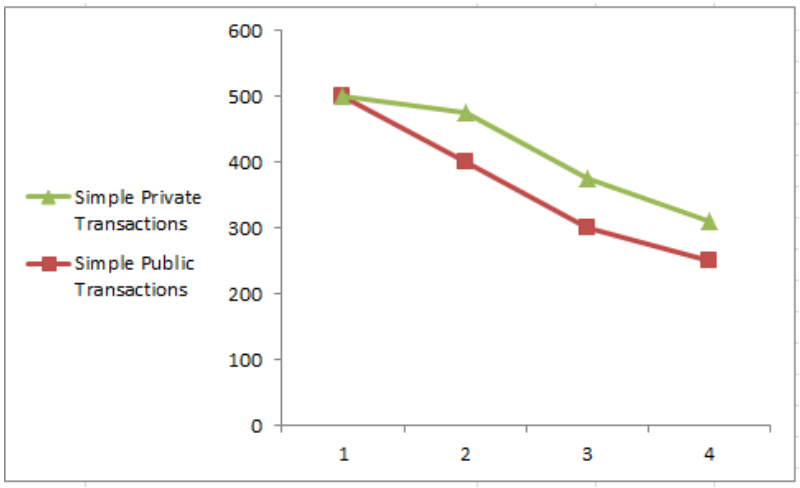

Figure 21. Effects of endorsement policy variation.

7.7.2. Network Architecture

If the number of peers per organization increases and the number of endorsers is kept constant, the maximum throughput increases. The same occurs if the number of organizations increases and the other two variables remain constant. For this setup, the optimum throughput was two peers per organization with one endorser. Therefore, employing the right peers could enhance throughput by up to 31% and 21% in public and private transactions, respectively.

7.8. Setup Configuration

7.8.1. Database Type

The performance of Hyperledger Fabric is subject to database choice. Although only CouchDB was used in the experiment, the results show that the performance of Fabric was relatively lower, especially for private transactions, than the literature-reported values for other databases.

7.8.2. Hardware

The system's performance will improve with better hardware. Given a small number of CPUs, an increase in their number is expected to boost throughput. The performance of CouchDB is higher with a greater number of CPUs. Moreover, CPU utilization declines with more cores. The CPU cannot be fully utilized in hardware with many cores, and no single core could achieve 90% CPU utilization. The hardware enhancement also reduces the number of crashes, and therefore, maintains system integrity.

7.8.3. Block Parameters

The ordering service creates new blocks at maximum block size or a definite period after creating a previous block. The maximum throughput can be kept below 500 tx/s by varying the block size to initiate the production of new blocks. A maximum block time below 2 s ensures at most 1000 tx within the maximum block time. Therefore, a lower block time yields a reduced throughput because new blocks' creation, transfer, and validation attract higher overhead costs. Figures 22–25 illustrate the changes in maximum throughput and latency with block time and size.

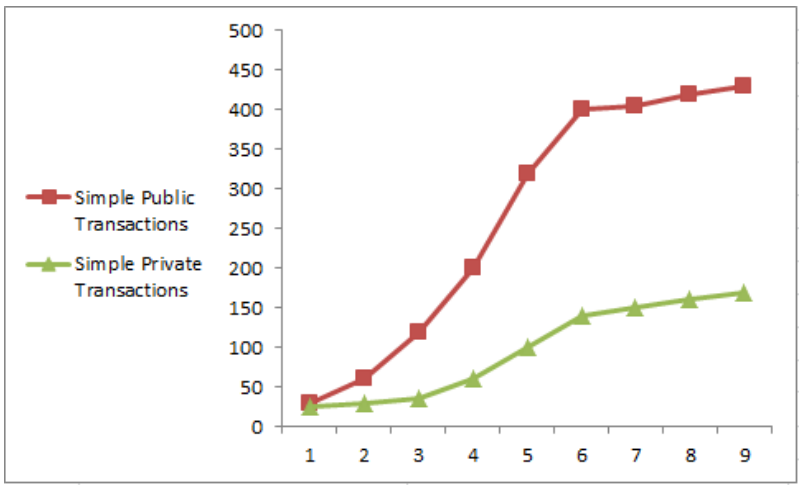

Figure 22. Maximum throughput vs. block time.

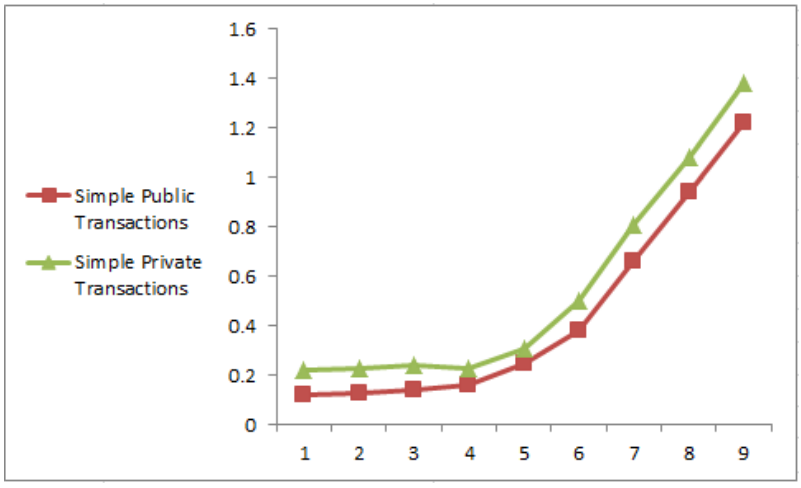

Figure 23. Latency at low throughput vs. block time.

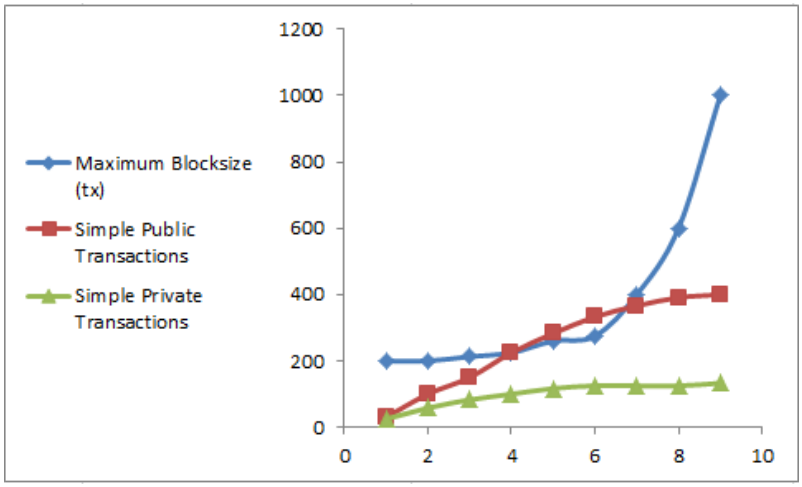

Figure 24. Maximum throughput vs. block size.

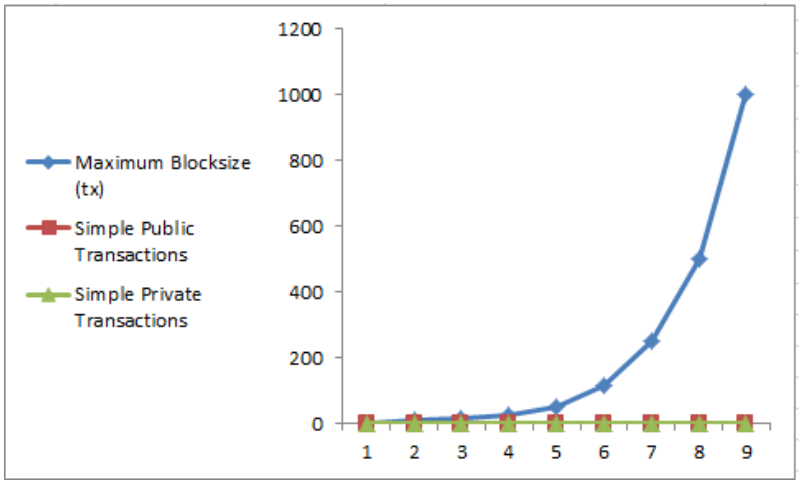

Figure 25. Latency at low throughput vs. block size.

7.9. Business Logic

7.9.1. I/O Heavy Workload

This is the impact of keeping large datasets concerning CouchDB's keyspace size. No relevant correlation was found for smaller keyspace sizes of less than 105. The maximum amount of sustainable throughput decreased as the amount of data per transaction increased. The drop was more drastic for public than private transactions.

After confirming that the keyspace size does not affect sizes lower than 105, a reading speed of 400 reads per second was achieved in CouchDB for simple key-based queries. Complex queries had 150 reads per second. Non-invoked queries were utilized because they are not part of Fabric's transaction flow.

7.9.2. CPU Heavy Workload

Matrix multiplications were utilized to evaluate Fabric's performance on CPU-demanding operations. Simple nested loops were utilized with various matrix sizes to achieve numeric control of complexity. The magnitude of operations tends to saturation as the size of matrices gets large. The results indicate that Fabric performed well under CPU-intensive tasks.

8. Discussion

The performance of the Hyperledger Fiber framework was evaluated to determine its suitability for delivering integrity and privacy-aware electronic health records systems in a blockchain. The experimental metrics utilized to achieve this objective included perfor-

mance, gauged throughput and latency, business logic, architecture, and setup variables. Anonymity and security evaluations involve the assessment of potential vulnerabilities of a permissioned distributed network and how risks can be minimized. The DLPS was used as a benchmarking solution, which proved effective with the choice of CouchDB as the database. In terms of architecture, the number of orderers did not affect the performance under a few transactions. However, the addition of peers under a constant endorsement policy enhances throughput.

The hardware and database setup is essential in ensuring optimal performance. Better hardware yields high performance when the number of CPUs is low. CouchDB performed slightly slower than the speeds reported for LevelDB in the literature for private and public transactions (Kuzlu et al.) [119]. In business logic, the performance declines considerably beyond a transaction payload of 1kB. The endorsement policy affects performance if redundancy calculations increase. The performance of reading and writing is independent of the index matrix. Based on these observations, it is also clear that throughput depends on the type and number of transactions, and the choice of database.

In this research, the author conducted a performance analysis for the completeness and acceptability of our proposed smart contract. In evaluating the capability of Fabric to support secure health record infrastructure, the performance of the framework was analyzed in terms of architecture, business logic, and setup configuration concerning our proposed health record system. The results demonstrate that Hyperledger Fabric is suitable and customizable for implementing a privacy-aware health records system in the blockchain. Since it is a permissioned platform, security policies are straightforward and focus on managing the human as the weakest link in any system. These results are echoed by other researchers who tested the Fabric with other databases, hardware platforms, and cloud-based facilities. We received almost a similar performance to the others in the blockchain network. We endorse that our proposed solution did not create an additional overhead on the Hyperledger Fabric.

9. Future Work

Plenty of work has been done on the privacy and security of health record systems. However, with the wide acceptance of blockchain networks in the IT industry and the decentralized network adoption, there was a need for a health record system that should be enabled on blockchain networks. The author discussed various blockchains in health-record-related solutions in the literature review, as seen in Section 3. However, to our knowledge, we have yet to find any user-patient-centric health record system with the blockchain and usage control model. We have completed our solution and have not tested this with the existing HRS system. There are some limitations to this research that need to be addressed in future work; we implemented the complete healthcare system within the blockchain networks as a smart contract. The problem with the existing HRS systems is that they are distributed and implemented in a unified fashion. To integrate our solution with the existing HRS system, there should be a generic and standard framework such as HL7. Thus, to bring our proposed work to the next level, we have to provided integration with HL7. Another consideration for future work is cross-validation, which is an important aspect of this research which should be performed in future.

10. Conclusions

In this paper, we proposed an Integrity and Privacy-Aware, Patient-Centric Health Record Access Control Framework using blockchain. We investigated the literature regarding the user-centric privacy of patients' healthcare data records on top of the blockchain; analyzed the famous blockchain operating systems to choose the optimized one; and described our proposed framework design based on blockchain technology using Hyperledger Fabric and usage control (UCON), implementation, and evaluation. Our contributions have been to achieve that it is user-centric and allows control by the patient to access his/her health record for the concerned health personnel. As a result of our evaluation and analysis,

the implemented prototype using Hyperledger Fabric is suitable, efficient, and customisable for implementing a privacy-aware health records system in the blockchain. Since it is a permissioned platform, security policies are straightforward.

Author Contributions: Supervision, T.A.S.; writing—original draft, R.A.A.; writing—review and editing, T.A.S. and S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Liu, H.; Crespo, R.G.; Martínez, O.S. Enhancing privacy and data security across healthcare applications using blockchain and distributed ledger concepts. In *Healthcare*; Multidisciplinary Digital Publishing Institute: Basel, Switzerland, 2020; Volume 8, p. 243.
2. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
3. Li, W.; He, M.; Haiquan, S. An Overview of Blockchain Technology: Applications, Challenges and Future Trends. In Proceedings of the 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC) 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 18–20 June 2021; pp. 31–39.
4. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
5. Hylock, R.H.; Zeng, X. A blockchain framework for patient-centered health records and exchange (HealthChain): Evaluation and proof-of-concept study. *J. Med. Internet Res.* **2019**, *21*, e13592. [CrossRef]
6. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *4*, 21260.
7. Ali, S.I.M.; Farouk, H.; Sharaf, H. A blockchain-based models for student information systems. *Egypt. Inform. J.* **2021**, *23*, 187–196. [CrossRef]
8. Yadav, A.S.; Agrawal, S.; Kushwaha, D.S. Distributed ledger technology-based land transaction system with trusted nodes consensus mechanism. *J. King Saud-Univ.-Comput. Inf. Sci.* **2021**, *34*, 6414–6424. [CrossRef]
9. Zhang, R.; Xue, R.; Liu, L. Security and privacy on blockchain. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–34. [CrossRef]
10. Shi, S.; He, D.; Li, L.; Kumar, N.; Khan, M.K.; Choo, K.K.R. Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. *Comput. Secur.* **2020**, *97*, 101966. [CrossRef]
11. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parallel Distrib. Comput.* **2019**, *134*, 180–197. [CrossRef]
12. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22 May 2016; pp. 839–858.
13. Meisami, S.; Beheshti-Atashgah, M.; Aref, M.R. Using Blockchain to Achieve Decentralized Privacy In IoT Healthcare. *arXiv* **2021**, arXiv:2109.14812.
14. Reyna, A.; Martín, C.; Chen, J.; Soler, E.; Díaz, M. On blockchain and its integration with IoT. Challenges and opportunities. *Future Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]
15. Ali, S.; Wang, G.; White, B.; Cottrell, R.L. A blockchain-based decentralized data storage and access framework for pinger. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1 August 2018; pp. 1303–1308.
16. Fouka, G.; Mantzorou, M. What are the major ethical issues in conducting research? Is there a conflict between the research ethics and the nature of nursing? *Health Sci. J.* **2011**, *5*, 3.
17. Hepp, T.; Sharinghousen, M.; Ehret, P.; Schoenhals, A.; Gipp, B. On-chain vs. off-chain storage for supply-and blockchain integration. *It-Inf. Technol.* **2018**, *60*, 283–291. [CrossRef]
18. Dubovitskaya, A.; Baig, F.; Xu, Z.; Shukla, R.; Zambani, P.S.; Swaminathan, A.; Jahangir, M.M.; Chowdhry, K.; Lachhani, R.; Idnani, N.; et al. ACTION-EHR: Patient-centric blockchain-based electronic health record data management for cancer care. *J. Med. Internet Res.* **2020**, *22*, e13598. [CrossRef] [PubMed]
19. Reijers, W.; Wuisman, I.; Mannan, M.; De Filippi, P.; Wray, C.; Rae-Looi, V.; Cubillos Vélez, A.; Orgad, L. Now the code runs itself: On-chain and off-chain governance of blockchain technologies. *Topoi* **2021**, *40*, 821–831. [CrossRef]

20. Khatoon, A. A blockchain-based smart contract system for healthcare management. *Electronics* **2020**, *9*, 94. [CrossRef]
21. Mackey, T.K.; Miyachi, K.; Fung, D.; Qian, S.; Short, J. Combating health care fraud and abuse: Conceptualization and prototyping study of a blockchain antifraud framework. *J. Med. Internet Res.* **2020**, *22*, e18623. [CrossRef]
22. Alansari, S. A Blockchain-Based Approach for Secure, Transparent and Accountable Personal Data Sharing. Ph.D. Thesis, University of Southampton, Southampton, UK, 2020.
23. Salman, T.; Zolanvari, M.; Erbad, A.; Jain, R.; Samaka, M. Security services using blockchains: A state of the art survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 858–880. [CrossRef]
24. Sharma, Y.; Balamurugan, B. Preserving the privacy of electronic health records using blockchain. *Procedia Comput. Sci.* **2020**, *173*, 171–180. [CrossRef]
25. Yaga, D.; Mell, P.; Roby, N.; Scarfone, K. Blockchain technology overview. *arXiv* **2019**, arXiv:1906.11078.
26. Hardjono, T.; Smith, N. Cloud-based commissioning of constrained devices using permissioned blockchains. In Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, Xi'an, China, 30 May 2016; pp. 29–36.
27. Ali, M.S.; Vecchio, M.; Pincheira, M.; Dolui, K.; Antonelli, F.; Rehmani, M.H. Applications of blockchains in the Internet of Things: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1676–1717. [CrossRef]
28. Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
29. Lahbib, A. Distributed Management Framework Based on the Blockchain Technology for Industry 4.0 Environments. Ph.D. Thesis, Institut Polytechnique de Paris, Paris, France, 2020.
30. Adler, J.; Berryhill, R.; Veneris, A.; Poulos, Z.; Veira, N.; Kastania, A. Astraea: A decentralized blockchain oracle. In Proceedings of the 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July 2018; pp. 1145–1152.
31. Ellis, S. A Decentralized Oracle Network Steve Ellis, Ari Juels, and Sergey Nazarov. 2017. Available online: https://academy.bit2me.com/wp-content/uploads/2021/05/chainlink-whitepaper.pdf (accessed on 1 March 2022).
32. Zhang, F.; Cecchetti, E.; Croman, K.; Juels, A.; Shi, E. Town crier: An authenticated data feed for smart contracts. In Proceedings of the 2016 aCM sIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24 October 2016; pp. 270–282.
33. Peterson, J.; Krug, J.; Zoltu, M.; Williams, A.K.; Alexander, S. Augur: A Decentralized Oracle and Prediction Market Platform (v2. 0). Whitepaper 2019. Available online: https://augur.net/whitepaper.pdf (accessed on 15 June 2022).
34. Missier, P.; Bajoudah, S.; Capossele, A.; Gaglione, A.; Nati, M. Mind my value: A decentralized infrastructure for fair and trusted iot data trading. In Proceedings of the Seventh International Conference on the Internet of Things, Linz, Austria, 22 October 2017; pp. 1–8.
35. Guarnizo, J.; Szalachowski, P. PDFS: Practical data feed service for smart contracts. In Proceedings of the European Symposium on Research in Computer Security, Luxembourg, 23–27 September 2019; pp. 767–789.
36. Ritzdorf, H.; Wüst, K.; Gervais, A.; Felley, G.; Capkun, S. *TLS-N: Non-Repudiation over TLS Enabling-Ubiquitous Content Signing for Disintermediation*; Cryptology ePrint Archive: Zurich, Switzerland, 2017.
37. de Pedro, A.S.; Levi, D.; Cuende, L.I. Witnet: A decentralized oracle network protocol. *arXiv* **2017**, arXiv:1711.09756.
38. Hess, Z.; Malahov, Y.; Pettersson, J. Æternity blockchain: The trustless, decentralized and purely functional oracle machine. In *White Paper*; 2017. Available online: https://blockchainlab.com/pdf/91ternity-blockchain-whitepaper.pdf (accessed on 15 June 2022).
39. Sztorc, P. Truthcoin. In *Peer-to-Peer Oracle System and Prediction Marketplace*; 2015. Available online: https://www.truthcoin.info/ (accessed on 1 March 2022).
40. Mo, B.; Su, K.; Wei, S.; Liu, C.; Guo, J. A solution for internet of things based on blockchain technology. In Proceedings of the 2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Singapore, 31 July–2 August 2018; pp. 112–117.
41. Liang, X.; Zhao, J.; Shetty, S.; Li, D. Towards data assurance and resilience in IoT using blockchain. In Proceedings of the MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 23–25 October 2017; pp. 261–266.
42. Hepp, T.; Wortner, P.; Schönhals, A.; Gipp, B. Securing physical assets on the blockchain: Linking a novel object identification concept with distributed ledgers. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 15 June 2018; pp. 60–65.
43. Pan, J.; Wang, J.; Hester, A.; Alqerm, I.; Liu, Y.; Zhao, Y. EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts. *IEEE Internet Things J.* **2018**, *6*, 4719–4732. [CrossRef]
44. Ahn, J. EdenChain: The programmable economy platform. *Eden Singap. White Pap.* **2018**, *1*, 13–44
45. Draskovic, D.; Saleh, G. Decentralized data marketplace based on blockchain. *White Pap.* **2017**.
46. Michelin, R.A.; Dorri, A.; Steger, M.; Lunardi, R.C.; Kanhere, S.S.; Jurdak, R.; Zorzo, A.F. SpeedyChain: A framework for decoupling data from blockchain for smart cities. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, New York, NY, USA, 5–7 November 2018; pp. 145–154.
47. Lombardi, F.; Aniello, L.; De Angelis, S.; Margheri, A.; Sassone, V. A Blockchain-Based Infrastructure for Reliable and Cost-Effective. In *IoT-Aided Smart Grids*; IET: London, UK, 2018

48. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasubramanian, V. Continuous patient monitoring with a patient centric agent: A block architecture. *IEEE Access* **2018**, *6*, 32700–32726. [CrossRef]
49. Kang, J.; Yu, R.; Huang, X.; Wu, M.; Maharjan, S.; Xie, S.; Zhang, Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet Things J.* **2018**, *6*, 4660–4670. [CrossRef]
50. Gao, Z.; Xu, L.; Turner, G.; Patel, B.; Diallo, N.; Chen, L.; Shi, W. Blockchain-based identity management with mobile device. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 15 June 2018; pp. 66–70.
51. Yang, H.; Liang, Y.; Yuan, J.; Yao, Q.; Yu, A.; Zhang, J. Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5G and beyond. *IEEE Trans. Ind. Inform.* **2020**, *16*, 7094–7104. [CrossRef]
52. Yang, H.; Bao, B.; Li, C.; Yao, Q.; Yu, A.; Zhang, J.; Ji, Y. Blockchain-enabled tripartite anonymous identification trusted service provisioning in industrial IoT. *IEEE Internet Things J.* **2021**, *9*, 2419–2431. [CrossRef]
53. Yang, H.; Liang, Y.; Yao, Q.; Guo, S.; Yu, A.; Zhang, J. Blockchain-based secure distributed control for software defined optical networking. *China Commun.* **2019**, *16*, 42–54. [CrossRef]
54. Marstein, K.E.H. Improve Auditing and Privacy of Electronic Health Records by Using Blockchain Technology. Master's Thesis, The University of Bergen, Bergen, Norway, 2019.
55. Antonopoulos, A.M. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2014.
56. Comstedt, E. *Increasing the Trust between Automotive Actors Using a Hyperledger Fabric Blockchain*; DiVA: Stockholm, Sweden, 2019.
57. Stinson, D.R. *Cryptography: Theory and Practice*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2005.
58. Thomsen, S.S.; Knudsen, L.R. Cryptographic Hash Functions. Ph.D. Thesis, 2005. Available online: https://backend.orbit.dtu.dk/ws/portalfiles/portal/5025771/sst_thesis_v1.0.pdf/ (accessed on 25 July 2022).
59. Feistel, H. Cryptography and computer privacy. *Sci. Am.* **1973**, *228*, 15–23. [CrossRef]
60. Merkle, R.C. Method of Providing Digital Signatures. US Patent 4,309,569, 5 September 1982.
61. Merkle, R.C. A certified digital signature. In Proceedings of the Conference on the Theory and Application of Cryptology, New York, NY, 20–24 August 1989; pp. 218–238.
62. Wang, L.; Shen, X.; Li, J.; Shao, J.; Yang, Y. Cryptographic primitives in blockchains. *J. Netw. Comput. Appl.* **2019**, *127*, 43–58. [CrossRef]
63. Schneider, F.B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv. (CSUR)* **1990**, *22*, 299–319. [CrossRef]
64. Buterin, V.; Griffith, V. Casper the friendly finality gadget. *arXiv* **2017**, arXiv:1710.09437.
65. Sompolinsky, Y.; Zohar, A. Secure high-rate transaction processing in bitcoin. In Proceedings of the International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, 26–30 January 2015; pp. 507–527.
66. Ichikawa, D.; Kashiyama, M.; Ueno, T.; et al. Tamper-resistant mobile health using blockchain technology. *JMIR MHealth UHealth* **2017**, *5*, e7938. [CrossRef]
67. Baset, S.A.; Desrosiers, L.; Gaur, N.; Novotny, P.; O'Dowd, A.; Ramakrishna, V. *Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer*; Packt Publishing Ltd.: Birmingham, UK, 2018.
68. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
69. Giammusso, S. Blockchain for Education Case Study on Hyperledger Fabric. Ph.D. Thesis, Politecnico di Torino, Torino, Italy, 2019.
70. Park, J.; Sandhu, R. The UCONABC usage control model. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2004**, *7*, 128–174. [CrossRef]
71. Sandhu, R.; Park, J. Usage control: A vision for next generation access control. In *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*; Springer: Berlin, Heidelberg, 2003; pp. 17–31.
72. Park, J.; Sandhu, R. Towards usage control models: Beyond traditional access control. In Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, Monterey, CA, USA, 3 June 2002; pp. 57–64.
73. Lazouski, A.; Martinelli, F.; Mori, P. Usage control in computer security: A survey. *Comput. Sci. Rev.* **2010**, *4*, 81–99. [CrossRef]
74. Zhang, X. *Formal Model and Analysis of Usage Control*; George Mason University: Fairfax, VA, USA, 2006.
75. Um-e Ghazia, R.M.; Shibli, M.A.; Bilal, M. Usage control model specification in xacml policy language. *Comput. Inf. Syst. Ind. Manag.* **2012**, *7564*, 68–79.
76. Katt, B.; Zhang, X.; Breu, R.; Hafner, M.; Seifert, J.P. A general obligation model and continuity: Enhanced policy enforcement engine for usage control. In Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, Estes Park, CO, USA, 11–13 June 2008; pp. 123–132.
77. Griggs, K.N.; Ossipova, O.; Kohlios, C.P.; Baccarini, A.N.; Howson, E.A.; Hayajneh, T. Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *J. Med Syst.* **2018**, *42*, 1–7. [CrossRef]
78. Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. Medrec: Using blockchain for medical data access and permission management. In Proceedings of the 2016 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, 22–24 August 2016; pp. 25–30.
79. Bell, L.; Buchanan, W.J.; Cameron, J.; Lo, O. Applications of blockchain within healthcare. *Blockchain Healthc. Today* **2018**, *2*, 130–139. [CrossRef]

80. Kuo, T.T.; Ohno–Machado, L. Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. *arXiv* **2018**, arXiv:1802.01746.
81. Dwivedi, A.D.; Srivastava, G.; Dhar, S.; Singh, R. A decentralized privacy-preserving healthcare blockchain for IoT. *Sensors* **2019**, *19*, 326. [CrossRef] [PubMed]
82. Zyskind, G.; Nathan, O.; et al. Decentralizing privacy: Using blockchain to protect personal data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 180–184.
83. Lee, T.F.; Chang, I.P.; Kung, T.S. Blockchain-Based Healthcare Information Preservation Using Extended Chaotic Maps for HIPAA Privacy/Security Regulations. *Appl. Sci.* **2021**, *11*, 10576. [CrossRef]
84. Ivan, D. Moving toward a blockchain-based method for the secure storage of patient records. In *ONC/NIST Use of Blockchain for Healthcare and Research Workshop*; ONC/NIST: Gaithersburg, MD, USA, 2016; pp. 1–11.
85. Chen, Y.; Ding, S.; Xu, Z.; Zheng, H.; Yang, S. Blockchain-based medical records secure storage and medical service framework. *J. Med Syst.* **2019**, *43*, 5. [CrossRef] [PubMed]
86. Brown, C.A.; Bailey, J.H.; Davis, M.E.M.; Garrett, P.; Rudman, W.J. Improving patient safety through information technology. *Perspect. Health Inf. Manag. Am. Health Inf. Manag. Assoc.* **2005**, *2*, 5.
87. White, F. Primary health care and public health: Foundations of universal health systems. *Med Princ. Pract.* **2015**, *24*, 103–116. [CrossRef]
88. Organization, W.H. *World Health Statistics 2018: Monitoring Health for the SDGs, Sustainable Development Goals*; World Health Organization: Geneva, Switzerland, 2018.
89. Ahmad, B.I.; et al. User acceptance of health information technology (HIT) in developing countries: A conceptual model. *Procedia Technol.* **2014**, *16*, 1287–1296.
90. Fielding, M.; Odero, B.; Ochieng, C. From paper to data: Taking medical health records into the future. In *Transforming Africa*; 2016; p. 136. Available online: https://www.researchgate.net/profile/Matthew-Fielding/publication/314283699_From_paper_to_data_taking_medical_health_records_into_thefuture/links/59311142aca272fc55e72e85/From-paper-to-data-taking-medical-health-records-into-the-future.pdf (accessed on 5 Augest 2022).
91. Fortino, G.; Badica, C.; Malgeri, M.; Unland, R. *Intelligent Distributed Computing VI: Proceedings of the 6th International Symposium on Intelligent Distributed Computing-IDC 2012, Calabria, Italy, September 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 446.
92. Alotaibi, Y.K.; Federico, F. The impact of health information technology on patient safety. *Saudi Med J.* **2017**, *38*, 1173. [CrossRef]
93. Filkins, B.L.; Kim, J.Y.; Roberts, B.; Armstrong, W.; Miller, M.A.; Hultner, M.L.; Castillo, A.P.; Ducom, J.C.; Topol, E.J.; Steinhubl, S.R. Privacy and security in the era of digital health: What should translational researchers know and do about it? *Am. J. Transl. Res.* **2016**, *8*, 1560.
94. Keshta, I.; Odeh, A. Security and privacy of electronic health records: Concerns and challenges. *Egypt. Inform. J.* **2021**, *22*, 177–183. [CrossRef]
95. Shirtawi, S.; Godla, S.R. Enhancing the Framework for E-Healthcare Privacy and Security: The Case of Addis Ababa. In *Innovative Data Communication Technologies and Application*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 499–516.
96. Capece, G.; Lorenzi, F. Blockchain and Healthcare: Opportunities and Prospects for the EHR. *Sustainability* **2020**, *12*, 9693. [CrossRef]
97. Steria, S. A blockchain-based healthcare platform for secure personalised data sharing. *Public Health Inform. Proc. MIE* **2021**, *281*, 208.
98. Mani, V.; Manickam, P.; Alotaibi, Y.; Alghamdi, S.; Khalaf, O.I. Hyperledger healthchain: Patient-centric IPFS-based storage of health records. *Electronics* **2021**, *10*, 3003. [CrossRef]
99. Shafiee, M.; Shanbehzadeh, M.; Kazemi-Arpanahi, H. Common data elements and features of brucellosis health information management system. *Inform. Med. Unlocked* **2022**, *30*, 100953. [CrossRef]
100. Wu, J.; Zhou, P.; Chen, Q.; Xu, Z.; Ding, X.; Hao, J. Blockchain-based Privacy-Aware Contextual Online Learning for Collabrative Edge-Cloud-Enabled Nursing System in Internet of Things. *IEEE Internet Things J.* **2021**, 1. [CrossRef]
101. Wang, S.; Gao, Y.; Sha, N.; Zhang, G.; Luo, H.; Chen, Y.; Zhang, L.; Wang, Y.; Fang, X.; Zhao, C.; et al. Technical characteristics and model of blockchain. In Proceedings of the 10th IEEE International Conference on Communication Software and Networks ICCSN 2018, Chengdu, China, 6–9 July 2018.
102. Stafford, T.F.; Treiblmaier, H. Characteristics of a blockchain ecosystem for secure and sharable electronic medical records. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1340–1362. [CrossRef]
103. Puthal, D.; Malik, N.; Mohanty, S.P.; Kougianos, E.; Das, G. Everything you wanted to know about the blockchain: Its promise, components, processes, and problems. *IEEE Consum. Electron. Mag.* **2018**, *7*, 6–14. [CrossRef]
104. Chenthara, S.; Ahmed, K.; Wang, H.; Whittaker, F. A novel blockchain based smart contract system for ereferral in healthcare: HealthChain. In Proceedings of the International Conference on Health Information Science, Amsterdam, The Netherlands, 20–23 October 2020; pp. 91–102.
105. Naresh, V.S.; Reddi, S.; Allavarpu, V.D. Blockchain-based patient centric health care communication system. *Int. J. Commun. Syst.* **2021**, *34*, e4749. [CrossRef]
106. Zheng, X.R.; Lu, Y. Blockchain technology–recent research and future trend. *Enterp. Inf. Syst.* **2022**, *16*, 12. [CrossRef]
107. Labazova, O.; Dehling, T.; Sunyaev, A. From hype to reality: A taxonomy of blockchain applications. In Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019), Maui, HI, USA, 8–11 January 2019.

108. Sabu, S.; Ramalingam, H.; Vishaka, M.; Swapna, H.; Hegde, S. Implementation of a Secure and privacy-aware E-Health record and IoT data Sharing using Blockchain. *Glob. Transit. Proc.* **2021**, *2*, 429–433. [CrossRef]
109. Barati, M.; Rana, O. Privacy-aware cloud ecosystems: Architecture and performance. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5852. [CrossRef]
110. Toyoda, I.; Nuno, F.; Shimizu, Y.; Umehira, M. Proposal of 5/25-GHz dual band OFDM-based wireless LAN for high-capacity broadband communications. In Proceedings of the 2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin, Germany, 11–14 September 2005; Volume 3, pp. 2104–2108.
111. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B. Performance analysis of hyperledger fabric platforms. *Secur. Commun. Netw.* **2018**, *2018*, 1–14. [CrossRef]
112. Baliga, A.; Solanki, N.; Verekar, S.; Pednekar, A.; Kamat, P.; Chatterjee, S. Performance characterization of hyperledger fabric. In Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, Switzerland, 20–22 June 2018; pp. 65–74.
113. Thakkar, P.; Nathan, S.; Viswanathan, B. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In Proceedings of the 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Milwaukee, WI, USA, 28 September 2018; pp. 264–276.
114. Javaid, H.; Hu, C.; Brebner, G. Optimizing validation phase of hyperledger fabric. In Proceedings of the 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Rennes, France, 21–25 October 2019; pp. 269–275.
115. Sharma, S.; Chen, K.; Sheth, A. Toward practical privacy-preserving analytics for IoT and cloud-based healthcare systems. *IEEE Internet Comput.* **2018**, *22*, 42–51. [CrossRef]
116. Dubovitskaya, A.; Urovi, V.; Vasirani, M.; Aberer, K.; Schumacher, M.I. A cloud-based ehealth architecture for privacy preserving data integration. In Proceedings of the IFIP International Information Security and Privacy Conference, Hamburg, Germany, 26–28 May 2015; pp. 585–598.
117. Roehrs, A.; Da Costa, C.A.; da Rosa Righi, R. OmniPHR: A distributed architecture model to integrate personal health records. *J. Biomed. Inform.* **2017**, *71*, 70–81. [CrossRef]
118. Guggenberger, T.; Sedlmeir, J.; Fridgen, G.; Luckow, A. An in-depth investigation of the performance characteristics of Hyperledger Fabric. *Comput. Ind. Eng.* **2022**, *173*, 108716. [CrossRef]
119. Kuzlu, M.; Pipattanasomporn, M.; Gurses, L.; Rahman, S. Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 536–540.