MedBloc: A Blockchain-based Secure EHR System for Sharing and Accessing Medical Data

Jack Huang, Yuan wei Qi, Muhammad Rizwan Asghar, Andrew Meads, and Yu-Cheng Tu
School of Computer Science, The University of Auckland, New Zealand
{xhua451, yqi955}@aucklanduni.ac.nz, {r.asghar, andrew.meads, yu-cheng.tu}@auckland.ac.nz

***Abstract*—In New Zealand, there is currently no shared Electronic Health Record (EHR) system integrated between major healthcare organisations, such as hospitals, medical centres, and specialists. Thanks to its characteristics, blockchain technology can be a suitable platform for building a large-scale EHR system for New Zealand. In this paper, we present MedBloc, a blockchain-based secure EHR system that enables patients and healthcare providers to access and share health records in a usable yet privacy-preserving manner. MedBloc captures a longitudinal view of the patient's health story and enables patients to give or withdraw consent for regulating access to their records. To protect medical data, MedBloc uses an encryption mechanism and enforces a smart contract based access control mechanism for regulating access. MedBloc not only demonstrates how the blockchain can establish New Zealand's first shared EHR system but it shows how blockchain can potentially disrupt the entire medical technology domain.**

***Index Terms*—EHR Protection, EHR Sharing, Secure Blockchain, NZ Healthcare System for New Zealand**

I. Introduction

The New Zealand healthcare sector has grown steadily in the last decade. From 2006 to 2016, the total number of discharges from publicly funded hospitals has on average increased by 2.79% annually [1]. Due to this additional burden on the existing medical infrastructure and services, the cost of medical treatment and the time to process patients are likely to increase as the New Zealand healthcare sector grows. In the face of an ever-increasing demand for healthcare services, healthcare organisations are looking for efficient and scalable IT solutions to improve productivity, cost of treatment, and quality of care in the New Zealand healthcare sector [2].

Currently, New Zealand has a highly diverse, non-unified health IT landscape [2]. There are 20 District Health Boards (DHBs) that provision health services in the different regions of New Zealand. Each DHB has adopted different IT systems with varying levels of maturity to process, store, and share electronic medical data. The poor interoperability between health IT systems and long response time to data access requests limits effective data sharing and management, causing patient's health records to be fragmented, rather than holistic [3]. As a result, patients often must repeat their health story whenever they visit a new General Practitioner (GP) or healthcare provider [4]. In addition to this, healthcare providers lack access to accurate health information at the point of care due to how health data are distributed across various data storage systems. If medical information needs to be transferred from one healthcare provider to another, the patient consent must be obtained. However, New Zealand lacks a centralised electronic consent management platform that can capture patient consent in a usable, flexible, and transparent manner [5].

To promote universality across medical IT systems, the New Zealand Ministry of Health (MoH) proposed establishing a nationwide integrated system for the purposes of sharing Electronic Health Records (EHRs) [4]. An EHR is formally defined as "a longitudinal electronic record of patient health information generated by one or more encounters in any care delivery setting".

In this paper, we present MedBloc, a secure EHR system that leverages blockchain technology. Blockchain technology can address the fragmentation problem with existing health IT solutions by providing a distributed yet interconnected data storage framework. With the use of the blockchain, there is no central authority managing all data. Instead, ownership remains with the patients who are empowered to access their EHRs and share them with healthcare providers on the network [6]. The immutable and transparent nature of data on the blockchain enables data auditability and provenance, allowing patients to see how their EHRs are used and enabling them to have a more active role in managing their health [3], [6]. Furthermore, a blockchain network has no single point of failure, offering continuous data availability that otherwise is difficult to achieve with centralised systems [7].

The main obstacle to implementing a shared EHR system on the blockchain is that, traditionally, data on the blockchain is completely transparent to all participants, thus making user privacy difficult to preserve [6]. Recent innovations in blockchain technology, such as the introduction of permissioned blockchain networks, offer improved privacy settings by only allowing selected parties to participate in the network [7], [8]. However, as transactions are still stored in plain text, an adversary can steal data by accessing the blockchain nodes or pretending to be a healthcare provider. Thus, data stored on the blockchain nodes must be secured with an encryption mechanism, and access control policies must be enforced to regulate access to sensitive data.

In recent years, several blockchain-based healthcare applications have also been proposed. Azaria et al. proposed MedRec, the first functional shared EHR system that leverages blockchain technology [3], but it raises confidentiality and privacy concerns. FHIRChain is a system similar to MedRec, which uses cryptographic techniques to encrypt and share data between clinicians [9]. Dubovitskaya et al. proposed a framework, also similar to MedRec [10]. It enables patients to share their health records but does not provide any way to manage consent revocation. MedShare uses blockchain for enforcing access control and monitoring of health data stored on the cloud [11]. However, it has scalability issues.

In MedBloc, we address the above issues by storing healthcare records immutably on the blockchain and having them encrypted so that no adversary, apart from the patient and their trusted healthcare providers, can view or update them. MedBloc employs a consent management system at a healthcare provider level so that patients can control which providers are able to access their records at any given time. This is achieved by using smart contracts enforcing access control policies. We also consider usability aspects in MedBloc. Important cryptographic material used to secure records or to share records is managed by an authentication server to prevent users from losing or exposing the material to unauthorised parties.

The core contributions of this paper are as follows.

- We present MedBloc, a fully fledged blockchain-based shared EHR system that empowers patients to securely share their medical data with healthcare providers, as well as offering the ability to capture patient consent.
- MedBloc protects data by employing encryption schemes and enables secure access to medical data.
- We propose a key storage framework that aims to improve usability by leveraging an authentication server for storing the cryptographic material. This brings convenience for the users of the system, as they do not have to be concerned about how they should keep their cryptographic material secure.

Although we target New Zealand healthcare sector in this work, without loss of generality, this work can be applied to healthcare sectors in other countries too.

The rest of this paper is organised as follows. In Section II, we give some background information, review related work, list down use case scenarios, and describe key requirements that our solution needs to fulfil. In Section III, we explain our system model, threat model, and proposed approach. Then, Section IV elaborates on the proposed solution. In Section V, we analyse the proposed solution from a security point of view. In Section VI, we evaluate the performance of the proposed solution. Section VII is dedicated to the discussion. Finally, we conclude in Section VIII.

II. Background and Requirements

A. New Zealand's Health IT Landscape

New Zealand's current health IT landscape is a diverse ecosystem of enterprise health applications, each with its own standards and level of maturity [2]. Data is spread over different health IT system, making EHR sharing inefficient and tedious. Typically, EHRs can be shared between patients and healthcare providers via proprietary portals. If referrals have to be made, electronic messages must be exchanged between the providers. GP-to-GP platforms can also be utilised to exchange records. As of 2015, the NZ Health IT Board maintains 25 national IT programmes. There is a need for DHBs to consolidate their systems so that the health sector can better collaborate and integrate data [2].

In 2016, the New Zealand MoH launched the New Zealand Health Strategy, with the goal of establishing a people-powered, smart health system by 2025 [4]. The Digital Health Work Programme 2020 was developed in response to this strategy. One of the core components of the programme is to create an EHR that collates a patient's health information into a single longitudinal "health story" accessible to all healthcare stakeholders [4].

There are many benefits of establishing a nationwide EHR system. For instance, patients do not need to repeat their health story whenever they visit a new healthcare provider. Health professionals can have more accessible information at the point of care, which can lead to better decision making and reduced potential misdiagnosis and errors. Additionally, such an EHR system can be used as a population health monitor, where important health data can be easily drawn from the population so that governments can make more informed investment decisions and create more effective public health policies and other healthcare initiatives.

A report [4] prepared by the MoH outlined the scope of establishing a shared EHR system. They stressed that the EHR must provide a single longitudinal view of an individual's health journey over time. Thus, the medical information stored on the EHR must be of value to health professionals at the point of care, such as allergies, medical procedures, and care plans. The EHR must also be people-powered by enabling patients to have a more active role in managing their health and moving away from a doctor-centric model of healthcare. Additionally, the EHR system must provide an API to support access from different endpoints (hospitals, medical centres), and population health data must be easily extracted and readily available for third parties (say government and researchers).

B. Related Work

In the context of applying blockchain to the healthcare domain, Kuo et al. explored the potential use cases and evaluated its benefits and pitfalls [6]. They identified that blockchain technology could be used to improve medical data management. Although it provides benefits such as security, privacy, and availability, it has the issue of confidentiality, scalability, and Sybil attack. However, the evaluation only considered permissionless blockchain and did not take the emerging permissioned blockchain technology into account. Brodersen et al. investigated how integrating permissioned blockchain technology into the existing healthcare IT investments can address many of the issues in the current healthcare IT paradigm surrounding security, data integrity, privacy, and identity [12]. Permissioned blockchain enables: 1) the creation of secured and trusted care records through the use of cryptographic techniques; 2) immutable identity, which leads to better transparency and non-repudiation; 3) auditing consent that empowers the patients. Gordon et al. highlighted the blockchain features that could enable patient-driven interoperability, such as access control rules, and listed several barriers and mitigation techniques [13]. For example, the size of modern health data, such as imaging data, can be enormous but could be partially mitigated by just storing a summary of the data, such as a short report.

For existing blockchain-based frameworks, Azaria et al. introduced MedRec, a shared EHR system that leverages permissionless blockchain technology [3]. This is the first functional prototype that has ever been proposed. The blockchain only stores access permissions and data pointers, and there are off-chain data repositories where EHRs are held. Moreover, it employs smart contacts to enforce data integrity and access control rules.

Another permissionless blockchain-based framework is FHIRChain, proposed by Zhang et al. [9]. FHIRChain is a framework centered around interoperability between clinicians. The blockchain stores data pointers and it employs public key cryptography for managing user identities. Each user has a public/private identity key pair used for digital signatures and data encryption. FHIRChain uses a token-based permission model to overcome the limitations of permissionless blockchain. Contents are encrypted with their public identity key so that only users with the correct private key can view them. To share data, the sender first signs the corresponding data pointer with her private signing key then encrypts it with the receiver's public identity key to obtain an encrypted token. When the receiver wishes to access the data, the receiver will decrypt the token with her private key and uses the sender's public signing key to verify that it was indeed provided by the sender.

Dubovitskaya et al. proposed a permissioned blockchain-based framework on managing and sharing records [10]. The records are stored in the providers' local database and as well as the cloud. The blockchain stores access permissions and data pointers. Each user has a public-private key pair for signing and another for encryption. Each patient also has a symmetric patient key that is used to encrypt her data. To share data with a doctor, the patient key is encrypted with the doctor's encryption public key and shared with the doctor. Each access permission is associated with a doctor and can be updated and changed by the patient.

Xia et al. proposed a blockchain-based system that provides access control and monitoring for medical data stored among cloud service providers called MeDShare [11]. MedShare logs actions performed on requested data and automatically revokes access to the data if an unauthorised action is performed, providing effective data behaviour monitoring. However, this system is not scalable and has very poor performance.

C. Use Case Scenarios

To explain our system requirements, we have come up with three use case scenarios. Later, in Section IV, we will illustrate MedBloc's underlying system process under each use case scenario. Consider that all medical staff in Auckland City Hospital, Christchurch Hospital, and Auckland Medical Centre (AMC) use a shared EHR system to manage patient data. We consider that Alice is an 18-year old girl who just moved to New Zealand from the UK for work. She lives in Auckland and has not enrolled with New Zealand's healthcare system.

1) Scenario 1: Registration: Soon after Alice moved to Auckland, she encountered a heart condition and went to AMC for treatment. In order to keep a record of this treatment and make treatments of future illnesses less costly, Alice wants to register with AMC.

2) Scenario 2: Record Access: After Alice was treated by AMC, she wishes to review the details of her condition and the medication that she was prescribed with.

3) Scenario 3: Consent: During the public holidays, Alice decided to travel to Christchurch. Unfortunately, she got ill again and was transported to Christchurch Hospital for treatment. To allow the doctors in Christchurch Hospital to learn the details about her existing condition, Alice needs to give Christchurch Hospital her consent to let them view her health records. However, after the treatment, Alice would also like to withdraw that consent.

4) Scenario 4: Update record details: After another visit to AMC, Alice reviewed her records again and found some errors in one of the records created by AMC. She would like the record to be corrected.

5) Scenario 5: Remove record: While amending the errors in one of Alice's records, the GP from AMC accidentally created a duplicated record for her and would like to delete it.

D. System Requirements

Based on the scope outlined by the MoH in their report [4], we created a set of requirements which our EHR solution must satisfy:

- **People-powered EHR.** The right given by a patient to let the doctors view her health records is referred to as patient consent. Our EHR system should empower the patients by enabling them to have a more active role in managing their health and provide them with a convenient platform to engage with the system. The stewardship of health records is handled by the patients so that they no longer need to request access to their records from their healthcare providers. In addition to this, the system should provide patients with the ability to give and withdraw consent based on which healthcare provider can access their health records.
- **Single Longitudinal View of Health Journey.** Data on the EHR system should not describe detailed (episodic) medical encounters. Instead, it should provide an overview of the patient's health condition over time and capture key information, which is of value to healthcare professionals at the point of care.

Along with the requirements set by the MoH, we also formed additional requirements based on our review of existing related work [3], [6], [9]–[11].

- **Security and privacy.** The system should preserve the confidentiality of the patients' health records and protect the privacy of the patients. This includes the enforcement of several access control rules. Additionally, even if an

attacker manages to obtain any data from the storage systems, the data should still remain protected.
- **Efficiency.** The system should be efficient enough so common operations such as accessing records and managing consent can be performed in a timely fashion.
- **Scalability.** The system should be scalable enough to deal with all the citizens.
- **Availability.** Patients and healthcare providers should always be able to access health records, where the system must have no single point of failure.

III. Our Idea

This section first presents our system and threat models, and then an overview of MedBloc, including how it addresses the requirements described in Section II-D and its underlying architecture.

A. System Model

Consider MedBloc as a blockchain network composed of multiple nodes. In our system model, we have the following entities:

- **Patient.** This entity is a person who has received medical treatment or could receive treatment in New Zealand. Patients can have records made for them, and they can query the blockchain for information regarding their medical history. However, they are required to decrypt data on their end after it is retrieved from the blockchain.
- **Healthcare Provider (HP).** An HP is any healthcare organisation in New Zealand – such as medical centres, hospitals, and specialists – rather than individual healthcare professionals. HPs are responsible for creating health records for patients. These records must be encrypted before they are inserted into the blockchain. HPs are able to retrieve data from the blockchain but data needs to be decrypted on the HPs' end. They are also able to update and delete data.
- **Healthcare Professionals.** Healthcare Professionals represent any medical staff directly involved in the treatment of patients. The roles of healthcare professionals can be but not limited to doctors, GPs, physicians, specialists, and nurses.
- **Network Administrator.** They maintain the blockchain nodes on the network. The network administrators are responsible for managing patient identities and ensuring the provision of most health services in their district, say in a DHB zone.
- **Blockchain Network.** The core infrastructure of the system. The network is made up of interconnected "peer" nodes with a shared ledger. Participating nodes are responsible for storing ledger data, executing smart contracts, validating transactions, and committing blocks onto the ledger.
- **Certification Authority (CA).** A CA issues digital certificates for participants on the blockchain using Public Key Infrastructure (PKI) [14]. These certificates represent the identities of entities participating in the blockchain and are attached to transactions made on the blockchain network. This allows recipients of the transaction to verify the sender's identity and the integrity of the message.
- **Authentication Server.** All other entities are traditional entities, while this is a new entity in our system. It stores the cryptographic material that will be used to authenticate users and allow them to connect to the blockchain network.
- **Service Client.** The client is the first point of contact for a user with the blockchain. It submits transactions to execute CRUD (Create, Read, Update, Delete) operations on the ledger. Users must provide their login details if they wish to access the client. Different clients are provided to each type of user in the system, each one offering a different set of functionalities. For instance, only the network administrator's client may register new patients on the blockchain.

B. Threat Model

The focus in the threat model is the protection of patient health data from unauthorised or improper access by attackers. The term 'access' refers to the operations of viewing, disclosing, modifying, and deleting data. Patient health data refers to patients' health records and personal details.

We assume that there are two types of attackers: external attackers and internal attackers.

- **External Attacker.** It refers to attackers who do not have any level of authority to access data in the blockchain, such as a person who does not have an identity in the network. Such attackers may attempt to gain access to health data using different methods. Attackers can compromise a set of nodes to gain access to data in the blockchain and mount a dictionary attack on encrypted health records, or take down the nodes using methods such as Denial of Service Attack (DoS) or Distributed DoS (DDoS).
- **Internal (or Insider) Attacker.** It refers to users with some level of authorised access to the blockchain, but want to perform operations that they are not entitled to. For example, in the context of the use case explained in Section II-D, a malicious HP from Wellington may want to view and modify health records created by AMC without Alice's consent.

We assume that there are at least as many nodes in the blockchain network as the number of DHBs across New Zealand. We also assume that the majority of the nodes are adequately protected. This is plausible as the nodes could be maintained by DHBs.

C. Proposed Approach

To address the requirements described in Section II-D, we introduce MedBloc, a shared EHR system built on top of blockchain.

Blockchain technology addresses the availability requirement by providing a network of nodes with no single point of failure. This allows the system to be fully operational at

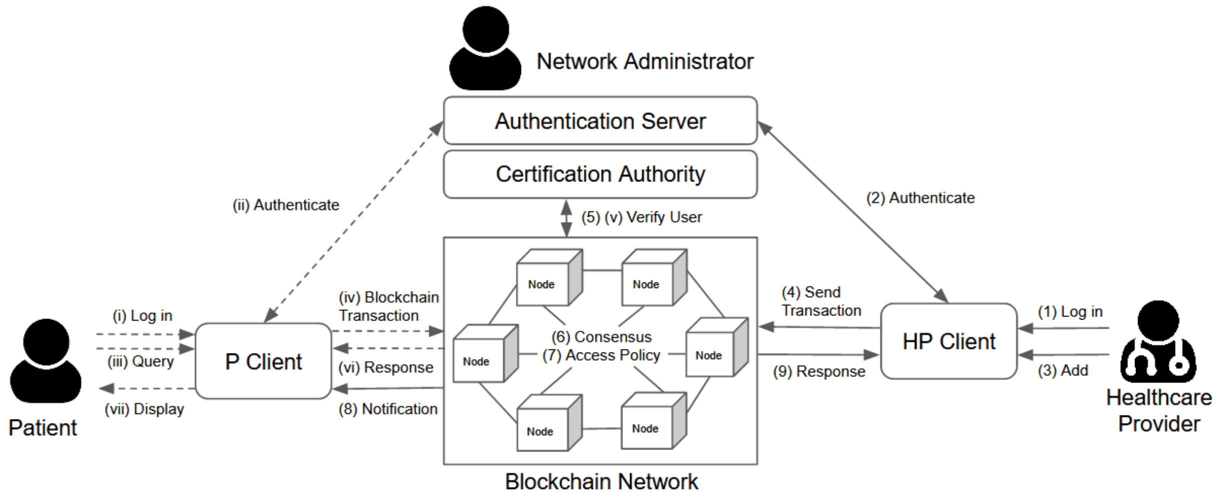

Fig. 1. The MedBloc Architecture. A 'P Client' is patients' client service, and an 'HP Client' is healthcare providers' client service. Network Administrators control the Authentication Server, the CA, and the overall Blockchain Network. The Blockchain Network is comprised of several nodes. Solid lines with arrows represent the system process (simplified) when an HP attempts to add a new record for a patient. Dotted lines with arrows represent the system process when a patient attempts to view her records.

all time as long as the majority of the nodes are active. While popular blockchain implementations, such as Bitcoin [15] and Ethereum [16], are known for their slow transaction processing speed (between 3-20 transactions per second [8]), our proposed system will use a permissioned blockchain network [7] [8] that employs a state machine replication-based consensus mechanism, allowing for comparatively faster performance and excellent scalability [17]. Since all data is transparent on the blockchain, we propose using non-traditional blockchain entities, such as authentication servers and Certificate Authorities, to provide means to issue identities and secure the cryptographic material, which will be used to encrypt all data on the blockchain. We will also use blockchain specific features, primarily smart contracts, to enforce access control rules. These techniques will help keep data confidential and safeguard patient's privacy. The use of smart contracts and cryptographic techniques will also allow our system to have the capability to facilitate EHR sharing and to capture patients' consent.

Fig. 1 illustrates MedBloc's architecture. The patient health records are encrypted and stored on the blockchain. Different users can interact with the blockchain, such as making transactions and querying the ledger, using a dedicated client service.

Consider scenario 1 (from Section II-C), where Alice has just registered with AMC. During registration, Alice's identity and cryptographic material are generated and are stored in a secure Authentication Server. This material includes a public-private key pair for secure record sharing and a symmetric key for efficient encryption of records. An encrypted copy of her symmetric key, along with her public key, is stored on the blockchain.

After treatment, Alice gives AMC her consent to add records for her and view future records. This will enable Alice's GP, Bob, to record details of her visit on the blockchain.

In order to do so, Bob must first login in on AMC's client using his username and password (1). An authentication server will verify his details and return AMC's identity and cryptographic material (2). After successful authentication, Bob will issue an “Add Record” transaction using the HP client (3). This will generate a Record object containing fields relevant to the type of diagnosis or treatment that was made. Table I illustrates an example of a completed Record object with fields relating to a medical condition.

All fields, except fields that identify the record and the entities involved (patient and healthcare provider), are encrypted using the symmetric key. Note that the encrypted symmetric key can only be retrieved if AMC has consent from Alice. In addition to the encrypted Record table, the transaction must also include the smart contract name (“Add Record”) and its version number. This transaction is signed with AMC's identity material and sent to any one of the peer nodes in the network (4).

When a peer node receives the transaction proposal, it will confirm the identity of the sender by verifying the signature on the proposal. It may contact the CA in order to do so. The blockchain network will initiate the consensus protocol after a specific number of proposed transactions are gathered or after a certain time period (6). The block generated after consensus is distributed to the peer nodes on the network.

Every peer node that has received the block will independently execute each transaction, using its specified smart contract, in the sequence in which it appears within the block. In the case of the adding records, access control policies are executed, as a part of the smart contract logic, to prevent HPs from viewing or creating records for patients who have not given them the consent to do so (7). Thus, when considering the current scenario, each peer node must check if AMC has

TABLE I
AN EXAMPLE RECORD FOR A HEALTH CONDITION.

ID	Date	Patient ID	HealthProvider	Reason-code	Reason-desc	Condition-start	Condition-end	Condition-code	Condition-desc
12	01/01/19	P#18e1..	HP#f0e4..	00000		01/01/19		AAAAA	Stroke

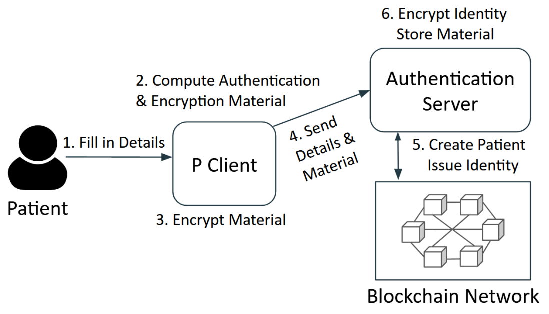

Fig. 2. Patient registration. Authentication and encryption material includes keys and other values that are needed for users to connect to MedBloc and decrypt or share records.

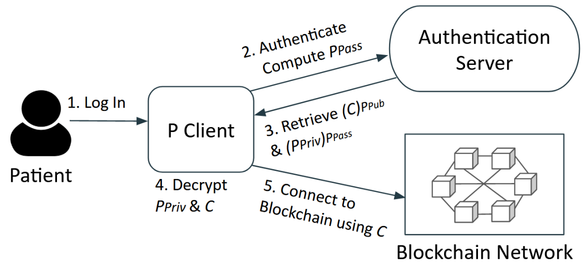

Fig. 3. Patient authentication.

Alice's consent to make records for her by looking through her "consent list" maintained on the blockchain ledger. Alice's consent list contains the HP identities that she has given consent to. The peer nodes will access this list and check if AMC's identity is on the list before it commits the record table onto the ledger.

Peers may emit an "event" after a specific transaction is committed. When Alice's record is committed, a peer node will notify Alice's client service of this event (8). It will also notify AMC's client service to indicate that the transaction has been successful (9).

Patients and HPs alike can issue transactions to query for data on the blockchain via their client services. For example, if Alice wishes to view her medical record after her recent visit to AMC, she must first log on to her patient client (i) and obtain her identity material via the authentication server (ii). She can then issue a request for her records using her client service (iii). This will invoke a "View Records" query, which is sent to the peer nodes (iv). Like the add record transaction, the peer nodes will first confirm her identity (v), then it executes the query on its ledger, which will retrieve all of Alice's records before sending them back to her client (vi). Note that consensus is not required for query-related transactions as the ledger does not need to be updated. Once data is retrieved from a peer node, the client will use Alice's symmetric key to decrypt the data so that it can display the results (vii).

IV. SOLUTION DETAILS

In this section, we explain the system process for the six key tasks that MedBloc can handle. These processes correspond to actions expected from a shared EHR system.

A. Registration

The registration process is depicted in Fig. 2 and is inspired by [18]. Consider scenario 1, where Alice wishes to be registered with MedBloc. To do so, she must first fill in her personal details on the patient client service (1). After all the necessary information is supplied, the client service will use the username and password pair to compute an authentication user key P_{Auth} and a verification value using some random user key parameters (2). It will also compute another user key for encryption using the same process, denoted as P_{Pass}. The client service will then generate a public-private key pair (P_{Pub}, P_{Priv}) and a symmetric patient key P_k, which would be used for secure record sharing and record encryption, respectively. The client will encrypt P_k with P_{Pub} to generate $(P_k)_{P_{Pub}}$, and P_{Priv} with P_{Pass} to create $\{P_{Priv}\}_{P_{Pass}}$ (3)[1]. Next, P_{Pub}, $(P_k)_{P_{Pub}}$, $\{P_{Priv}\}_{P_{Pass}}$, and the user key parameters (excluding password) are uploaded onto the authentication server as a registration request (4).

When the authentication server receives Alice's registration request, it will create a patient participant object on the blockchain using information in the request. The ID of the patient participant object will be Alice's health ID, and the participant object will contain Alice's personal detail, P_{Pub}, and $(P_k)_{P_{Pub}}$. The authentication server will form an identity for Alice by issuing a digital certificate for her via the CA and use that identity to create a network identity card (5) [19]. This network identity card contains an identity private key and an identity public key which is certified by the CA. This network identity card, denoted as C, will allow Alice to connect to MedBloc's blockchain network and make/sign transactions with the issued identity. The identity card will be encrypted with Alice's public key P_{Pub} to create $(C)_{P_{Pub}}$. The user key parameters, verification value, $(C)_{P_{Pub}}$, and $\{P_{Priv}\}_{P_{Pass}}$ will be stored on the authentication server (6).

B. Authentication

The authentication process is depicted in Fig. 3. Consider scenario 2, if Alice wishes to access her records, she must first log in to MedBloc using the patient client service (1). After entering the username and password, the client will

[1]Note that round brackets (.) represent asymmetric encryption and curly brackets {.} represent symmetric encryption.

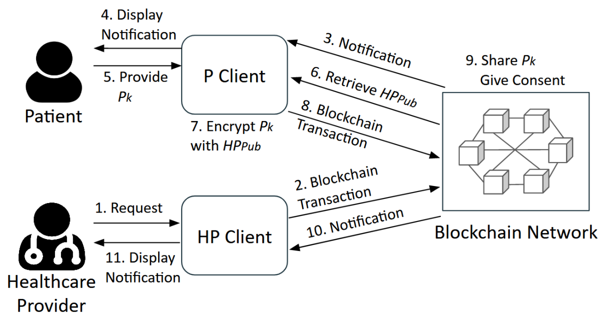

Fig. 4. Sharing records.

contact the authentication server and authenticate Alice using CompactPAKE [18], an asymmetric protocol that does not allow the authentication server to learn anything about the entered password except whether the password is correct or not. During the authentication process, the authentication key P_{Auth} will be computed for authenticating with the server, and the password key P_{Pass} will also be computed (2). After successful authentication, the authentication server will send the encrypted identity card and Alice's private key to the client service, which are denoted as $(C)_{P_{Pub}}$ and $\{P_{Priv}\}_{P_{Pass}}$, respectively (3). The client service decrypts $\{P_{Priv}\}_{P_{Pass}}$ with P_{Pass} then uses P_{Priv} to decrypt $(C)_{P_{Pub}}$ (4). The decrypted identity card, C, can then be used to connect to MedBloc's blockchain network (5). When an HP attempts to log in to MedBloc, the same process occurs.

C. Sharing Records

The process to share records is shown in Fig. 4 and is inspired by [20]. Consider scenario 3, where Alice would like to share important health details with the specialist she sees at Christchurch Hospital (CH). In order for CH to view and add records for a particular patient, they must have access to Alice's patient key P_k, which is used to encrypt all records for a patient. CH can request for Alice's P_k using their client service (1). When the request is made, CH's client will direct the request to the blockchain network's peer nodes as a transaction (2). When the transaction is executed, the peer nodes will send a notification event to other clients (3). The corresponding patient's client, which subscribes to *RequestKey* event, will pick up the message. This request will show up as a notification on Alice's client (4).

If Alice wishes to grant consent to CH to let them access her records, she must share her patient key P_k with CH using the *ShareKey* transaction through the patient client service (5). First, the client app will retrieve the requesting healthcare provider's (CH) public key, HP_{Pub}, from the blockchain (6). HP_{Pub} will be used to encrypt P_k, denoted as $(P_k)_{HP_{Pub}}$. $(P_k)_{HP_{Pub}}$ will be sent in a ShareKey transaction to the blockchain network (8). As part of the smart contract logic, a new PatientKey asset that contains $(P_k)_{HP_{Pub}}$ is created on the blockchain ledger, and a reference to CH's identity is added to Alice's consent list (9).

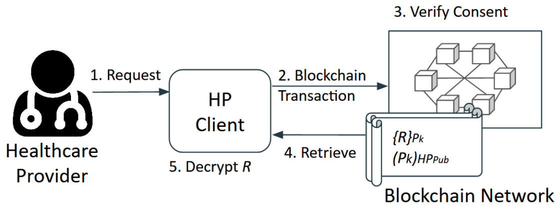

Fig. 5. Accessing records.

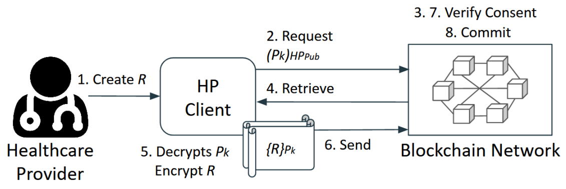

Fig. 6. Adding records as a Healthcare Provider.

After $(P_k)_{HP_{Pub}}$ is committed onto the ledger, and the consent list is successfully updated, the peer node will trigger a ShareKey event (10). CH's client will pick up on the event and then show a notification on the client to indicate that Alice has shared the key with them (11). CH will now know that they can access Alice's records.

D. Accessing Records

After successful authentication, whether the user is an HP or patient, their client service will have their private key, which is necessary for accessing records. Consider scenario 3, if CH wishes to access Alice's records, using the HP client service, after Alice has granted them consent, they can send a request (1) for Alice's records and her encrypted patient key, $(P_k)_{HP_{Pub}}$, as a transaction to the blockchain nodes via their client (2). The smart contracts on the blockchain will verify that Alice has listed the identity of CH in her consent list (3). After successful verification and smart contract execution, $(P_k)_{HP_{Pub}}$ and the encrypted records, $\{R\}_{P_k}$, is sent to CH's client (4). The client will decrypt $(P_k)_{HP_{Pub}}$ using HP_{Priv} to retrieve P_k. P_k will then be used to decrypt $\{R\}_{P_k}$. Finally, the decrypted records, R, will be displayed for viewing. This process is shown in Fig. 5. For scenario 2, a similar process occurs when Alice attempts to access her records. The smart contract will not check for consent but instead, just use Alice's identity to retrieve every record that she owns.

E. Adding Records

The process for adding records is illustrated in Fig. 6. In scenario 1, after Alice has received treatment, a record of her visit must be logged on the blockchain by AMC. AMC must first creates a new record using the HP client service (1). After all information is supplied, the client will then attempt to retrieve the encrypted patient key (belonging to Alice), $(P_k)_{HP_{Pub}}$, from the ledger (2). The smart contact on the blockchain will verify that CH is included in Alice's consent list (3) before it returns $(P_k)_{HP_{Pub}}$ to the client (4). Using

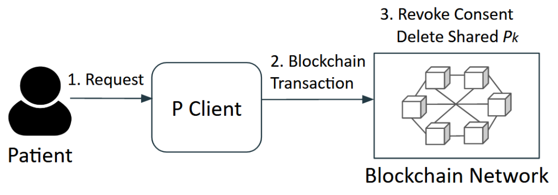

Fig. 7. Revoking sharing of records as a patient.

CH's private key, the client decrypts $(P_k)_{HP_{Pub}}$ to obtain (P_k), so it can encrypt the new record R (5). The encrypted record $\{R\}_{P_k}$ is sent to the blockchain as a transaction (6), where it will be checked by the smart contract again (7) and eventually committed onto the ledger as a record for Alice (8).

F. Revoke Sharing

Again consider scenario 3, where Alice wishes to revoke her consent to CH after she sees the specialist. Alice can revoke her consent using the patient client service (1). When doing so, a *Revoke Medical Records Sharing (RMRS)* transaction is created by their client service and sent to the blockchain (2). Once the blockchain received the transaction, the identity of CH, referenced in the RMRS transaction, is removed from Alice's consent list and the corresponding PatientKey asset $(P_k)_{HP_{Pub}}$ is deleted from the ledger (3). This process is shown in Fig. 7.

G. Updating Records

In scenario 4, Alice can request the record to be corrected by manually contacting AMC. Currently, there is no system in place to facilitate this process, so it can only be done through conventional methods such as email or phone. However, in the future, we may add a record update request transaction that the patients can send from their client service to the HPs. The process of updating a record is a combination of the process of accessing a record and add a record. To update the record as per Alice's request, AMC must first retrieve the record using the same process as in subsection IV-D. AMC will then update the record on the HP client service and send the encrypted and updated record to the blockchain using the same process as in subsection IV-E but as an update transaction. Consent will be checked again by the smart contract before the newly updated record is commited to the ledger. Notice that the old record will still exist in the blockchain due to its immutability, but the state database will only show the newest version [21].

H. Deleting Records

Consider scenario 5, to delete a record, the GP from AMC can simply issue a delete transaction on the duplicated record using the HP client service. The smart contract will verify consent as usual before deleting the record. Due to the immutable property of blockchain, data on the blockchain cannot really be deleted; the transaction will only mark the duplicated record as deleted so that it is removed from the world state database. The logically deleted record will still exist in the blockchain but cannot be used for any purposes.

V. Security Analysis

This section assesses how MedBloc mitigates the two fundamental threats defined in the threat model.

An external attacker may compromise one or more nodes in the network in an attempt to gain access to the data in the blockchain or take down/over the network. In our approach, all health data in the blockchain is encrypted to ensure confidentiality, so even if the attacker manages to obtain the data from the compromised nodes, without the encryption keys, which are only held by patients, the attacker would not be able to decipher the data. Our approach to encrypt each patients health data with a unique key also makes it virtually impossible to decrypt all the data. For encryption, we used well-known symmetric and asymmetric cryptographic primitives, namely, RSA and AES. Our solution is secure as long as these cryptographic primitives are secure. Without loss of generality, we can use new symmetric and asymmetric cryptographic primitives, which are more secure or efficient.

If the attacker's intention was to take down the system using methods such as DDoS, the attacker would need to shut down majority of the nodes in the network. As we assumed that there would be many nodes, attempting to attack the majority would be extremely difficult due to the computational required. Also, due to the distributed nature of blockchain technology, as long as the majority of nodes are still maintaining the network, the system would be continuously available. If the attacker's intention was to take over the network and alter the data in the blockchain, the attacker would need to compromise and control over 50% of the nodes in the network. If the attacker controls less than 50% of the nodes, the consensus process will ensure that data integrity is preserved.

Concerning insider attackers, we achieve security through access control and the consent mechanism. For instance, when the malicious healthcare provider from Wellington attempts to access Alice's health data, the smart contract would enforce the access control rules and check whether that healthcare provider is in Alice's consent list. Since Alice never gave consent to that healthcare provider, the healthcare provider would be denied access and prevented from accessing or altering Alice's health data. Without the patient key shared from Alice, even if the healthcare provider manages to obtain Alice's data from the blockchain, the healthcare provider would not be able to decrypt it.

VI. Performance Analysis

In this section, we analyse the performance of transaction reads and writes on a blockchain network that could support EHR sharing. In particular, we benchmark the impact of transaction size and access control rules on the overall network performance. To do so, we implemented a fully operational blockchain network using the Hyperledger Fabric platform [21]. Hyperledger Fabric is a permissioned smart contract based blockchain platform designed for enterprise use. It does not employ Proof-of-Work (PoW) consensus [6], [7], commonly known as *mining*. Instead, it relies on the Practical Byzantine Fault Tolerant [22] (PBFT) algorithm as its main

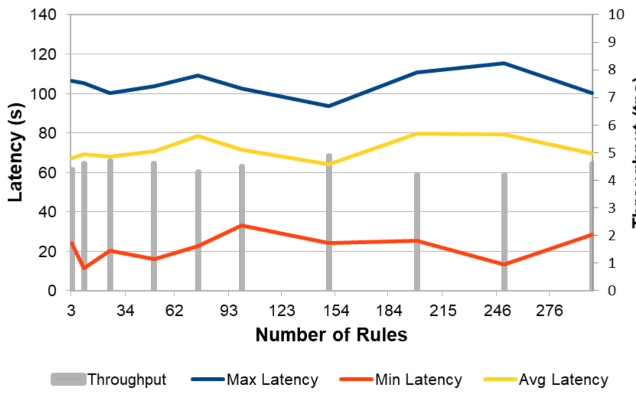

Fig. 8. Impact of the number of access control rules on performance.

consensus mechanism. This means that nodes do not have to expend as much energy as the ones in PoW-based blockchain platforms, such as Bitcoin [15] or Ethereum [16].

TABLE II
THE DEFAULT BENCHMARK CONFIGURATIONS.

Parameter	**Value**
File Size	4 kB
Block size	30 Tx
Block timeout	1 s
Send rate	30 tps
Database	CouchDB

Table II illustrates the default configuration of the Fabric network for all experiments unless stated otherwise.

Our Fabric network consisted of 3 organisations, each with 2 endorsing peers for a total of 6 peer nodes. Each peer node maintains the state of its ledger using CouchDB [23]. All nodes run within their own Docker container on a single machine with an Intel Core i5-3570 CPU @ 3.40GHz and 16GB of memory. The version of Fabric we used is v1.1.

Thakkar et al. [24] have conducted extensive performance benchmarks of the Hyperledger Fabric platform, as well as making optimisation recommendations. This includes observations on the impact of transaction arrival rate, block size, endorsement policy, channels, and ledger database on performance. Since they have extensively benchmarked the performance of Hyperledger Fabric, our experiments focus on the effect that access control policies and cryptography have on performance. Like their experiments, we focused on throughput and latency as the primary performance metrics in our benchmarks. Throughput is the rate at which transactions are committed to the ledger, measured in transactions per second (tps). Latency is the time taken from application sending the transaction proposal to the when the transaction is committed on the ledger.

A. Impact of Access Control Policies

Access Control policies are enforced by the smart contracts. For this test, in order to determine the effect caused by the number of access control rules on transaction processing performance, we implemented numerous arbitrary access policy rules. A minimum of three rules, relating to administrative privileges, are required for the smart contract to be deployed.

Fig. 8 shows that the number of access control rules has a negligible effect on the latency and throughput of the blockchain network when submitting records to the blockchain. The average latency of a blockchain network with 300 access control rules performed similarly as a network with the minimum number of rules. However, since we created arbitrary rules of relatively low complexity, it may not be reflective of the complex access control policies, which will need to be enforced in real world scenarios.

B. Impact of file size

The effect of the file size on the network performance will determine what type of medical information are suitable for the blockchain without causing network bottlenecks. A robust shared EHR system should have the ability to share relatively large files such as medical imagery and care plans. A blockchain network that has difficulty processing large files will have limited usage in the medical domain.

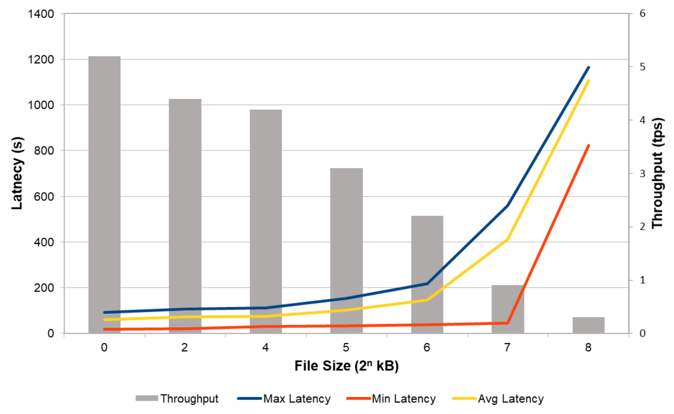

Fig. 9. Impact of file size on performance using CouchDB.

Fig. 9 shows the average, maximum and minimum throughput achieved when submitting records of different sizes to the blockchain. We noticed that latency grew exponentially with increasing file size. Between 1kB to 16kB, the average latency remained below 100s, with a throughput of 4-5tps. However, after this point, average latency grew to 146.18s at 64kB, before jumping exponentially to 413s and 1105.33s, at 128kB and 256kB, respectively.

While the size of a single text-based record is usually less than 64KB, if images are included in record transactions, transaction latency on MedBloc's network can get severely throttled due to larger transaction file sizes.

We also ran the same benchmark but using GoLevelDB instead of CouchDB. Fig. 10 illustrates the results. Throughput was more than double the corresponding CouchDB times across all file sizes up to 64kB, with the average latency being significantly lower as well. The overall latency trend

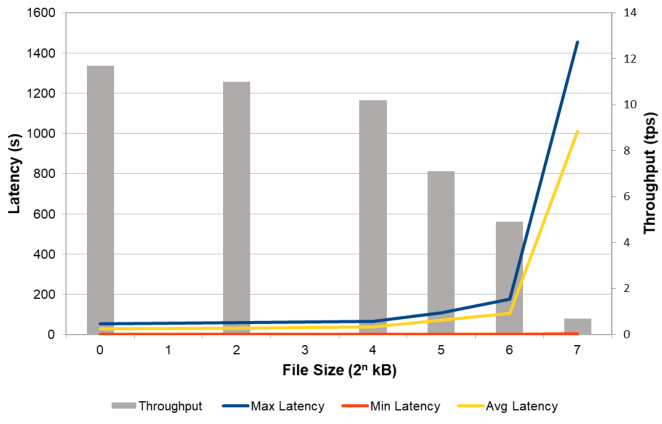

Fig. 10. Impact of file size on performance using GoLevelDB.

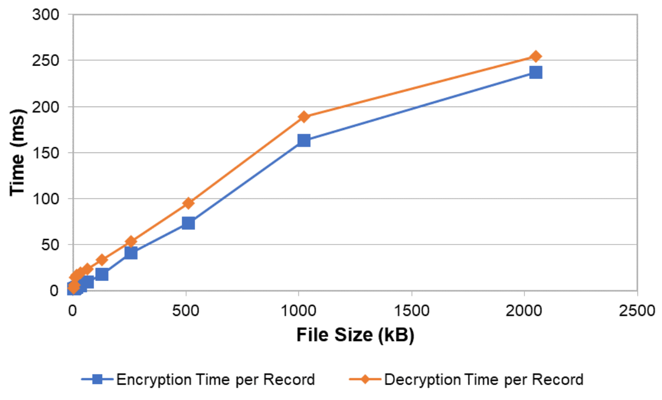

Fig. 12. CryptoJS AES encryption/decryption performance.

is exponential and is significantly worse than the equivalent time for CouchDB.

Faster latency and greater throughput were expected for LevelDB as it is well optimised for key-value stores. However, we found it surprising, that at higher file sizes, performance degraded more significantly than CouchDB.

C. Select Query Performance

Select queries are executed by a peer node when a patient or healthcare provider wishes to view records on the blockchain. The most computationally expensive operation of fetching records is that peer nodes must verify the identity of the sender and execute the access control policies over the transaction.

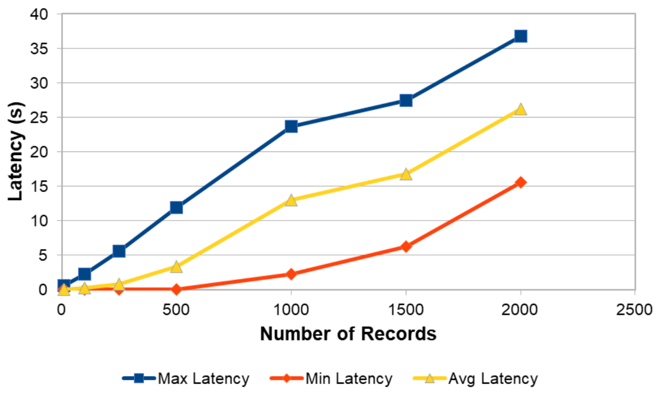

Fig. 11. Impact of the number of records fetched on performance.

The results in the previous section show that access control policies have a negligible effect on transaction processing speeds (throughput). Our experiments with select queries (illustrated in Fig. 11) further support this phenonmena. We fixed the arrival rate at 100 tps, with each transaction containing a select query for that will return a certain number of records. Note that our benchmarks were conducted on nodes configured with CouchDB databases as GoLevelDBs do not support batch reads, which are required for this test.

When we fetched less than 500 records per transaction (50000 records per second), we observed that the throughput remained near-level with the transaction arrival rate (100 tps). After this point, latency increased/throughput decreased linearly with the number of records that are fetched per transaction. At 2000 records/tx, the maximum latency never surpassed 35 seconds. When we consider realistically how much records a patient has (<500), records can be retrieved from the blockchain near instantly. This is especially helpful in medical emergency scenarios, where vital medical information must be accessed immediately at the point of care.

D. Encryption and Decryption

The time taken to encrypt or to decrypt records is important when considering that healthcare provider must access medical information near instantly at the point of care. We have chosen to measure the performance using CryptoJS, a JavaScript cryptography library [25]. Despite Javascript's reported vulnerabilities, we decided to use JavaScript as it is a quick prototyping environment for web apps. Our client app is likely to be a web app so that it can be easily accessed by anyone on any device (desktop, mobile) via the web browser. Thus, we found it would be most appropriate to use a JavaScript library to provide the cryptography functions.

Performance analysis of cryptographic libraries shows that CryptoJS offers relatively fast AES encryption/decryption times on both desktop and mobile devices and across a variety of browsers [20]. Thus, we used CryptoJS in our experiment. However, any other JavaScript-based encryption library is suitable for the browser-based client app.

Our experiments were conducted on the same machine as the one we used for the blockchain benchmarks. The tests were conducted on Google Chrome with Ubuntu 18.04. As expected, Fig. 12 shows AES encryption and decryption times (per record) increased linearly with increasing file size. Decryption took longer than encryption at all file sizes. For a 1MB record, decryption took on average 188.8 ms. However, if we consider a patient with a reasonably large collection of records (e.g., 500), we could be looking at a total decryption time of around 95 seconds, and this is excluding time taken to fetch the records from the blockchain.

Note that we report some blockchain related results in the appendix.

VII. Discussion

A. Fulfilling Requirements

MedBloc provides a people-powered EHR system, where patients retain primary stewardship over their health data and are able to have their consent captured. By giving each patient a key to encrypt their records, HPs must request permission from the patients to access their records, while patients are able to retrieve their health information at any time. Furthermore, the introduction of smart contract enforced access control provides a reliable and convenient way for patients to grant or revoke consent.

The limited types of record assets that could be stored on MedBloc (allergies, conditions, medications, etc.) restrict health information to only those that are valuable at the point of care. This reduces the number of detailed health records that need to be managed by the blockchain and thus improving the system's scalability. A longitudinal view of a patient's health story is captured on MedBloc. Health records are stored immutably on the blockchain, which means that healthcare professionals can see a patient's progression over time since all records generated remain on the ledger permanently and are ordered chronologically.

MedBloc provides patients and healthcare providers easy access to health records. It's decentralised network allows any blockchain node to serve the needs of the patient or healthcare provider. Each blockchain node provides the same functionality (same smart contract code), shares the same ledger, and can still commit transactions even if not all nodes in the network are running. Thus, by storing health records directly on the blockchain, it means that health records are continuously available on the blockchain. The level of data redundancy achieved by MedBloc is difficult and costly to replicate on the typical database system.

B. Managing Authentication Material

In order for a user to connect to MedBloc and decrypt health records, the user would need his identity card and private key, which can be called as the authentication material. Having an authentication server to store them is necessary as, without it, the users would have to manage the storage of the materials themselves, which does not provide good usability and can compromise security. It is also very likely that the user might not always have the material with them when they need them. With the solution we proposed in Section IV, which utilises authentication server, we provided a secure and convenient way for users to connect to MedBloc, as the users only have to remember their username and password.

Although our approach to authentication is secure, there is also the matter of how the authentication servers should be implemented and who should host them. This is critical to MedBloc's security as this may affect how resilient they are to attacks. To this end, we have considered two different approaches.

One approach is to have the healthcare providers provide the authentication servers. HPs' authentication servers will only store the authentication material for patients that are registered with them and a patient's authentication request will only be handled by the HP that the patient is registered with. For example, considering scenario 2, since AMH is Alice's registered HP, only AMH will store the authentication material for Alice, no one else. When Alice wants to authenticate, the request will only be handled by the authentication servers from AMH.

Another approach is to have a second logical blockchain just for user authentication. This sub-blockchain uses the same peer nodes from the main blockchain but is logically independent. It stores all the materials required to connect to the blockchain and handles all the authentication requests from the users.

These two approaches have different benefits and drawbacks. For instance, the implementation of the former approach would be simpler than the latter, as the authentication servers do not need to interact with another. Having the authentication servers separated from the blockchain also means that HPs could scale their servers based on the number of patients they have and since the servers do not need to validate transactions, it is potentially more scalable. The latter approach, on the other hand, provides better availability and security. One problem with the former approach is that if an HP's authentication server fails, any patients registered with that HP will not be able to connect to the blockchain. Since the latter approach leverages blockchain, the authentication service will be available as long as there are nodes running. The former approach is also not as secure as the latter since if an attacker compromises an authentication server, the attacker would be able to modify the contents in the server and potentially the tamper with the authentication process. However, because that all authentication materials are encrypted, it would difficult for an attacker to obtain the materials in plain text.

C. Consent List Vulnerability

Currently, to assess whether an HP can access a patient's health records, the smart contract must verify that the identity of the HP is in the patient's consent list. To allow this process to happen, the patient's consent list must be public so that it can be accessed and checked by any HP. Therefore, it could be possible for an HP to obtain the identities of all the HPs that the patient has given consent to from the list, which would be a serious security risk. However, this may not matter as only the smart contract can perform the consent check and will only verify the existence of the HP's identity in the list. We could simply disallow manual access by the HP's to mitigate this vulnerability.

An approach that can address this vulnerability is to encrypt every identity in the list with the corresponding HP's public key. To check whether an HP has been given consent by a patient, the smart contract will attempt to decrypt every identity in the list using the HP's private key. If the smart contract finds one that can be successfully decrypted and is the HP's identity, then that means that the patient has given the HP consent. However, this could hurt performance in determining access, so it might not be ideal. Alternatively, we can use the encrypted patient key created during the consent process as an access token. The smart contract will check whether an encrypted patient key exists between an HP and a patient to determine whether the patient has given that HP consent.

D. Special Scenarios

In this paper, we have only explored some of the most typical scenarios in which MedBloc could be used. There are many special scenarios which MedBloc does not consider. For instance, what happens when a patient does not want to give consent to a healthcare provider, but still wants to be treated and have a record of the treatment? One possible approach is to allow the HP to encrypt the record in the patient's public key and add it to the blockchain. When the patient logs in next time, the smart contract could re-encrypt the records with the patient key.

Another special scenario is when a patient is incapacitated and therefore unable to give consent to the HP performing the treatment. One possible approach to solve this problem is to introduce a transaction that gives a healthcare provider temporary view access to a patient's health records. However, it has the potential to be abused and could compromise the confidentiality of patients' data.

E. Fine-grained Access Control

For the sake of simplicity, access control in MedBloc is currently very coarse-grained. Once a patient gives consent to an HP, that HP would be able to access all records of that patient and that patient cannot control which records the HP has access to. In order to make access control in MedBloc more fined grained, an approach that we have considered is to introduce file keys and the notion of group, inspired by [20].

Instead of encrypting all records for a patient with the patient key, we can encrypt each record with a unique symmetric key specific to that file, called the file key. The file keys are encrypted with the patient key and stored alongside their corresponding record on the blockchain. In the context of the use case, if Alice wishes to give Auckland City Hospital (ACH) access to only one of her records, she would just need to share the file key for that record. To do this, she could make a new transaction called *ShareFileKey*. This transaction will add ACH's identity to Alice's consent list, retrieve and decrypt the file key for the shared record, encrypt it with ACH's public key, and then add it to the blockchain as a FileKey asset, similar to sharing a patient key. ACH will be then able to retrieve the encrypted record and file key, decrypt the file key with their private key and use that file key to decrypt the record. Because ACH does not have the file keys for Alice's other records, ACH will not be able to access them. This means that Alice will have control over record access on per record basis. If Alice still wishes to give ACH access to all of her records, she can still use the ShareKey transaction to share her patient key.

In many occasions, multiple HPs will work together to treat a patient. For instance, a patient with cancer may be treated by different hospitals and specialists at different locations for an extended period of time. In such a case, the patient and the HPs associated with the treatment can be considered as a shared group. A shared group represents a group of HPs treating a patient for a particular purpose that can range from days to years. A shared group has its own public and private key used to encrypt the file keys of the records created during the treatment. This allows the group to keep the records associated with the treatment private unless explicitly shared by the patient. For example, Alice may create a shared group with AMC and a cardiology service center called Auckland Heart Group (AHG) for the treatment of her heart condition. Each record created by this group will have its file key encrypted in the groups' public key and stored in another asset on the blockchain called GroupKey, similar to PatientKey and FileKey. The records between Alice, AMC, and AHG cannot be accessed by other HPs such as ACH unless Alice explicitly shares the records with them.

Overall, this approach gives us three access levels including access to all records, access to group records, and access to individual records. However, this granularity may come with a cost, such as the increased difficulty for a patient to manage consent and worse performance due to more information needing to be stored and communicated.

From the side of healthcare providers, since an HP participant represents an entire healthcare organisation, any staff in the organisation might have all privileges of that participant. However, there are different types of healthcare professionals, and they should have different access rights. A nurse, for example, maybe be able to read patient health records but not add or modify them. One possible approach to this problem is to make every medical personnel a participant, but more information and identities would need to be stored.

F. Password Reset, Lost, and Recovery

To change the password, a process similar to the one in [18] will be followed. The Private key will be simply re-wrapped in a new password key. However, if a patient forgets her password, the patient will not be able to recover it. This is because, without the password, the client service will not be able to compute the password key which is necessary for retrieving the identity card and private key. With no way to decrypt the identity card and private key, the patient will not be able to connect to the blockchain and retrieve/decrypt records, meaning that all of that patient's records will be lost. To allow password reset, an entity must keep a backup of the identity card and private key so that they can be re-wrapped in a newly computed password key and uploaded to the authentication server. To address this issue, we could provide an option to allow patients to decide whether they want to keep a backup of the private key by themselves or by trusted authorities such as the DHBs or MoH.

G. Comparison with Existing Solutions

Unlike MedBloc, MedRec leverages Ethereum [3], a permissionless blockchain. It only provides pseudo-anonymity which could compromise confidentiality. The approach to not store the records on-chain is potentially more scalable, but it comes with the cost of availability as if a provider's database goes down, no records from that provider can be accessed. Also, the providers still retain primary stewardship of the records.

In comparison to FHIRChain [9], MedBloc does not require records to be signed as each record asset on the blockchain already contains the identity of the associated patient and healthcare provider. Also, since FHIRChain focuses on interoperability between clinicians, it does not address patient consent like MedBloc.

MedBloc has very similar consent and access control mechanism to [10], which also captures patient consent. However, we differ in the sense that we also provide a way for patients to revoke consent and the patient key. Dubovitskaya et al. also did not mention how the keys will be stored. We provide a usable password-based authentication scheme where the keys are encrypted and stored securely on the authentication server and the blockchain.

VIII. Conclusions and Future Directions

In this paper, we presented MedBloc, a blockchain-based shared EHR system aiming to unify New Zealand's diverse health IT landscape, that can address many of the problems in NZ's current healthcare system. We showed that MedBloc is a fully fledged, people-powered, shared EHR system that allows patients and healthcare providers to access and share health records conveniently through a dedicated client service. Through the use of smart contracts and cryptographic techniques, patients can give and withdraw consent at anytime. Healthcare providers can request for consent and add records which are encrypted and stored securely on the blockchain. Using the sophisticated encryption scheme, MedBloc also protects patients' privacy. MedBloc's immutable access control policies prevent any unauthorised actions from being performed.

We also provided a secure and user-friendly storage for the key material, the authentication server, so that patients only have to remember their password. MedBloc only stores health information that is valuable at the point of care, which minimises the volume and size of records that need to be managed by the blockchain. Furthermore, by using a permissioned blockchain, which does not require traditional computational expensive protocols for consensus, the latency and throughput are significantly improved. These factors make our system very efficient and scalable. By storing the records in the blockchain and allowing the numerous DHBs to maintain the nodes, the system will be resistive to attacks and continuously available.

In the future, we would like to implement more fine-grained access control, the mechanism to account for special scenarios, and password recovery using the approaches that we have discussed. We would like to benchmark our system at large scale to fully simulate a production environment.

References

[1] NZ Ministry of Health. (2018) Publicly funded hospital discharges - 1 july 2015 to 30 june 2016. [Online]. Available: https://www.health.govt.nz/publication/publicly-funded-hospital-discharges-1-july-2015-30-june-2016

[2] Deloitte, "Independent review of new zealand's electronic health records strategy," 2015.

[3] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 2016, pp. 25–30.

[4] C. Reid and G. Osborne, "Strategic assessment: Establishing the electronic health record," 2016.

[5] M. R. Asghar, T. Lee, M. M. Baig, E. Ullah, G. Russello, and G. Dobbie, "A review of privacy and consent management in healthcare: A focus on emerging data sources," in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, 2017, pp. 518–522.

[6] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1211–1220, 2017.

[7] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. IEEE, 2017, pp. 557–564.

[8] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 182–191.

[9] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "Fhirchain: Applying blockchain to securely and scalably share clinical data," *Computational and structural biotechnology journal*, vol. 16, pp. 267–278, 2018.

[10] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," in *AMIA Annual Symposium Proceedings*, vol. 2017. American Medical Informatics Association, 2017, p. 650.

[11] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14 757–14 767, 2017.

[12] C. Brodersen, B. Kalis, C. Leong, E. Mitchell, E. Pupo, A. Truscott, and L. Accenture, "Blockchain: Securing a new health interoperability experience," *Accenture LLP*, 2016.

[13] W. J. Gordon and C. Catalini, "Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability," *Computational and structural biotechnology journal*, vol. 16, pp. 224–230, 2018.

[14] J. Weise, "Public key infrastructure overview," *Sun BluePrints OnLine, August*, pp. 1–27, 2001.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[16] V. Buterin, "Ethereum white paper: A next-generation smart contract and decentralized application platform," 2014.

[17] M. Vukolić, "The quest for scalable blockchain Fabric: Proof-of-work vs. BFT replication," in *International workshop on open problems in network security*. Springer, 2015, pp. 112–125.

[18] C. Zeidler and M. R. Asghar, "AuthStore: Password-based authentication and encrypted data storage in untrusted environments," in *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)*, vol. 00, Aug 2018, pp. 996–1001. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TrustCom/BigDataSE.2018.00140

[19] "Business network cards." [Online]. Available: https://hyperledger.github.io/composer/latest/playground/id-cards-playground

[20] G. Gheorghe, M. R. Asghar, J. Lancrenon, and S. Ghatpande, "SPARER: Secure cloud-proof storage for e-health scenarios," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, Aug 2016, pp. 444–453.

[21] Hyperledger Fabric. (2019). [Online]. Available: https://hyperledger-fabric.readthedocs.io

[22] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[23] Apache Software Foundation. Couchdb. [Online]. Available: http://couchdb.apache.org

[24] P. Thakkar, S. Nathan, and B. Viswanathan, “Performance benchmarking and optimizing Hyperledger Fabric blockchain platform,” in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2018, pp. 264–276.

[25] E. Vosberg. [Online]. Available: https://github.com/brix/crypto-js