MedFabric4Me: Blockchain Based Patient Centric Electronic Health Records System

by

Manish Vishnoi

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved April 2020 by the
Graduate Supervisory Committee:

Dragan Boscovic, Co-Chair
Kasim Selcuk Candan, Co-Chair
Maria Grando

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

Blockchain technology enables a distributed and decentralized environment without any central authority. Healthcare is one industry in which blockchain is expected to have significant impacts. In recent years, the Healthcare Information Exchange(HIE) has been shown to benefit the healthcare industry remarkably. It has been shown that blockchain could help to improve multiple aspects of the HIE system.

When Blockchain technology meets HIE, there are only a few proposed systems and they all suffer from the following two problems. First, the existing systems are not patient-centric in terms of data governance. Patients do not own their data and have no direct control over it. Second, there is no defined protocol among different systems on how to share sensitive data.

To address the issues mentioned above, this paper proposes MedFabric4Me, a blockchain-based platform for HIE. MedFabric4Me is a patient-centric system where patients own their healthcare data and share on a need-to-know basis. First, analyzed the requirements for a patient-centric system which ensures tamper-proof sharing of data among participants. Based on the analysis, a Merkle root based mechanism is created to ensure that data has not tampered. Second, a distributed Proxy re-encryption system is used for secure encryption of data during storage and sharing of records. Third, combining off-chain storage and on-chain access management for both authenticability and privacy.

MedFabric4Me is a two-pronged solution platform, composed of on-chain and off-chain components. The on-chain solution is implemented on the secure network of Hyperledger Fabric(HLF) while the off-chain solution uses Interplanetary File System(IPFS) to store data securely. Ethereum based Nucypher, a proxy re-encryption network provides cryptographic access controls to actors for encrypted data sharing.

To demonstrate the practicality and scalability, a prototype solution of Med-

Fabric4Me is implemented and evaluated the performance measure of the system against an already implemented HIE. Results show that decentralization technology like blockchain could help to mitigate some issues that HIE faces today, like transparency for patients, slow emergency response and better access control.

Finally, this research concluded with the benefits and shortcomings of MedFabric4Me with some directions and work that could benefit MedFabric4Me in terms of operation and performance.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Dragan Boscovic for the continuous work to support my thesis study and related research. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master's study, He has not only given me experience with research, but also helped me expose myself to Industry and always pushed me for excellence.

I would like to thank Dr. Kasim Selcuk Candan and Dr. Adela Grando for their relentless support and guiding me throughout my thesis work. Dr. Adela's experience in Healthcare industry helped this project tremendously. I would like to thank Chainrider for helping through Hyperledger Fabric setup. I would like to thank Microsoft Corporation for providing resources on Microsoft Azure that supported deployment and testing blockchain systems. I would also like to thank Nucypher community for guiding me through system.

I would like to thank all members of the Blockchain Research Lab at ASU: Raj Sadaye, Kaushal Mhalgi, Sourabh Siddharth, Kiran Suresh, Sasa Pesic, Francis Mendoza for their contribution in discussions regarding consortium blockchains. I would specially like to thank Raj for his mentorship in Hyperledger Fabric.

Finally, I must express my very profound gratitude to my parents for providing me the unfailing support and continuous encouragement throughout my years of study and through the process of researching. This accomplishment would have not been possible without them.

Thank you.

Manish Vishnoi

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 INTRODUCTION 1

1.1 Types of Blockchain 3

1.2 Problem Statement 4

1.3 Hyperldger Fabric 6

1.3.1 Hyperledger Fabric Transaction Flow 10

1.4 Nucypher 11

1.4.1 Proxy Re-encryption(PRE) 12

1.4.2 Advantages of Nucypher 13

1.5 IPFS 15

1.6 Merkle Tree 15

1.7 Why a Decentralised Healthcare System ? 16

1.8 Hypothesis 20

1.9 Organization of Thesis 22

2 RELATED WORK AND RESEARCHES 23

2.1 Health Information Exchange - healthcurrent.org 23

2.2 MedRec Prototype in EHR and Medical Research 24

2.3 Ancile: Privacy-Preserving Framework for Access Control and Interoperability of EHRs 25

2.4 Secure and Trustable Electronic Medical Records Sharing using Blockchain 26

2.5 Comparison of MedFabric4Me with State of Art Systems 27

CHAPTER Page

3 METHODOLOGY 28
3.1 Hyperledger Fabric 28
3.2 Tamper Proof Off-Chain Storage 29
3.3 Cryptographic Scheme 31
3.4 System Design 32
3.4.1 Creating an EHR Record 35
3.4.2 Updating an EHR Record 37
3.4.3 Deleting an EHR Record 38
3.4.4 Sharing an EHR Record 39
3.5 MedFabric4Me System Deployment in Current HIE Setting 41
4 EXPERIMENTS AND RESULTS 42
4.1 Description of Experiments 42
4.1.1 Comparison Matrix 42
4.2 Actors in MedFabric4Me 43
4.3 Experimental Data 44
4.3.1 Description of EHR Data Management 44
4.3.2 Managing Records 45
4.4 Hyperledger Fabric System 45
4.4.1 Hyperledger Caliper 47
4.4.2 Performance Testing 48
Testbed 48
Orderer: Kafka 50
Peer Size 20 51
Adding Asset to Blockchain 51

Querying Ledger 52
Results 52
Peer Size 40 54
Adding Asset to Blockchain 54
Querying Ledger 54
Results 55
Peer Size 50 57
Adding Asset to Blockchain 57
Querying Ledger 57
Results 58
Orderer: RAFT 60
Peer Size 20 61
Adding Asset to Blockchain 61
Querying Ledger 61
Results 62
Peer Size 40 64
Adding Asset to Blockchain 64
Querying Ledger 64
Results 64
Peer Size 50 67
Adding Asset to Blockchain 67
Querying Ledger 67
Results 68
4.5 IPFS 70

CHAPTER Page

4.5.1 Benchmarking IPFS : js-ipfs Profiling 70
4.5.2 Performance Testing 70
Testbed 70
Performance Testing 71
Uploading Speed: 71
Downloading Speed: 72
Results 73
4.6 Nucypher 73
4.6.1 Timeit 74
Testbed 74
Performance Testing 74
Results 74
5 CONCLUSION 76
5.1 Performance Summary 76
5.2 HIE vs MedFabric4Me 78
5.2.1 Use Case Comparison Between MedFabric4Me and HIE 80
5.3 Future Work 83
REFERENCES 85

LIST OF TABLES

Table Page

4.1 Fixed Parameters During Performance Testing 49
4.2 Number of Peers During Testing 49
4.3 Performance Results for Adding Assets; 20 Peers Network 52
4.4 Performance Results for Querying Assets; 20 Peers Network 53
4.5 Performance Results for Adding Assets; 40 Peers Network 55
4.6 Performance Results for Querying Assets; 40 Peers Network 56
4.7 Performance Results for Adding Assets; 50 Peers Network 58
4.8 Performance Results for Querying Assets; 50 Peers Network 59
4.9 Performance Results for Adding Assets; 20 Peers Network 61
4.10 Performance Results for Querying Assets; 20 Peers Network 62
4.11 Performance Results for Adding Assets; 40 Peers Network 64
4.12 Performance Results for Querying Assets; 40 Peers Network 65
4.13 Performance Results for Adding Assets; 50 Peers Network 67
4.14 Performance Results for Querying Assets; 50 Peers Network 68
4.15 IPFS Uploading Speed 71
4.16 IPFS Downloading Speed 72
4.17 Nucypher Decryption Time 75
5.1 Per Month Records(Adding Assets) According to TPS 79

LIST OF FIGURES

Figure		Page
1.1	Hyperledger Fabric Architecture[2]	7
1.2	Hyperledger Fabric Transaction Flow[19]	10
1.3	Centralised KMS vs Proxy Re-encrytion KMS	12
1.4	Merkle Tree[8]	16
2.1	MedRec System Orchestration[11]	25
3.1	EHR Record Creation with Steps Numbered	36
3.2	EHR Record Update with Steps Numbered	37
3.3	Basic Sharing Structure	40
4.1	Hyperledger Caliper Architecture[1]	48
4.2	Kafka Hyperlerdger Structure	51
4.3	Hyperledger Fabric Kafka Network with 20 Peers; Adding Asset Performance	51
4.4	Hyperledger Fabric Kafka Network with 20 Peers; Query Performance	54
4.5	Kafka Network with 40 Peers; Adding Asset Performance	54
4.6	Hyperledger Fabric Kafka Network with 40 Peers; Query Performance	57
4.7	Hyperledger Fabric Kafka Network with 50 Peers; Adding Asset Performance	57
4.8	Hyperledger Fabric Kafka Network with 50 Peers; Query Asset Performance	60
4.9	RAFT Hyperlerdger Structure	60
4.10	Hyperledger Fabric Network with 20 Peers and RAFT Ordering Service	63
4.11	Hyperledger Fabric Network with 40 Peers and RAFT Ordering Service	66
4.12	Hyperledger Fabric Network with 40 Peers and RAFT Ordering Service	69
4.13	IPFS Uploading Speed	72

Figure Page

4.14 IPFS Downloading Speed 73
4.15 Nucypher Encryption and Decryption Performance 75
5.1 Kafka vs RAFT (Adding Asset) 76
5.2 Kafka vs RAFT (Adding Asset) 77
5.3 IPFS Uploading Speed 79

Chapter 1

INTRODUCTION

In 2008, Satoshi Nakamoto - an individual or group of individuals - released a paper that described bitcoin, a first of its kind, peer-to-peer cryptocurrency system, with blockchain as an underlying technology.

A blockchain is, in the simplest of terms, a database that records a time-stamped series of immutable records called blocks that are linked through cryptography. The blockchain has no central authority and is managed by a cluster of nodes not owned by a single entity.

For organizations, blockchain holds the guarantee of transactional transparency – the capacity to make secure, real-time communication networks with partners around the globe to help everything from supply chains to payment systems to healthcare services and medicinal services information sharing.

These are the core blockchain architecture components:

- **Node** - user or computer within the system that holds a copy of the ledger.
- **Transaction** - the smallest building block of a blockchain system that stores information in the form of append-only. Transactions in blockchain are immutable.
- **Block** - a block is a kind of data structure that combines multiple transactions and distribute them over nodes.
- **Chain** - a sequence of blocks in a specific order

- **Consensus (consensus protocol)** - a set of rules and arrangements that all nodes agree and define operations of blockchain

The economic, political, humanitarian and legal system benefits[22] of blockchain technology starts to make this clear that this is an extremely disruptive technology. A prime business driver for blockchain technology is to achieve greater transparency and substantiate the accuracy of transaction data across the digital information ecosystem.

The development of blockchain applications over the years can be broken down into three generations. The reason this break was created was that the blockchains that defined blockchain 2.0 offered significantly different use cases and opportunities than the previous generations.

1. **Blockchain 1.0:** The first generation is a simple cryptocurrency, due to lack of smart contracts, their applications were limited to currency transfers and digital payment systems.

2. **Blockchain 2.0:** The subsequent age is the expansion of blockchain 1.0 into privacy, identity management and the development of non-native asset blockchain tokens and capabilities with the help of smart contracts. Blockchain now starts to impact industries like stock markets, healthcare, real estate, science, etc...

3. **Blockchain 3.0:** Extending the logic to next-generation, Blockchain would have to offer a significant change. This includes personal data economy, cheap transaction costs(Lightning Networks), interoperability and complete compute protocols built on top of the blockchain that allows for truly decentralized autonomous organizations.

1.1 Types of Blockchain

The advancement in blockchain technology and expansion into different industries came up with new challenges, like data governance, privacy, Identity Management, etc. Depending on the use and requirements, there are two kinds of blockchains, Private and Public blockchain. Be that as it may, there are a few varieties as well, similar to Consortium and Hybrid blockchains[24].

- **Public Blockchain:** Public Blockchain: A public blockchain is a non-restrictive, permission-less distributed ledger system. There are no restrictions over nodes to join the network and perform operations like accessing records, verifying transactions or do proof-of-work, and do mining. The data on the public transaction chain is still safe because the transaction has been verified by the node and is immutable The most common applications of public blockchains are exchanging cryptocurrencies and create open networks. Bitcoin and Ethereum are popular examples of public blockchains.

- **Private Blockchain:** A private blockchain is an access-controlled or permission blockchain operative only in a closed network. Private blockchains are usually used by companies or organizations, and they can control the nodes or users participating in the blockchain. Participant organization controls the level of security, identity management and accessibility to assets. Therefore, in terms of operation and usability, a private blockchain is similar to the public blockchain in terms of use, but the network is small and restrictive. Some applications of private blockchain networks are asset tracking, supply chain management, freight reconciliation, digital identity, asset ownership, etc.

- **Consortium Blockchain:** A consortium blockchain is similar to a private blockchain, but rather one organization controlling the entire network, several organizations can participate in consortium blockchain. Organizations can be divided according to their function or use in the blockchain, and control their respective identity management and accessibility. Consortium blockchains by healthcare sectors, the energy sector, banks, government organizations, etc.

- **Hybrid Blockchain:** A hybrid blockchain is unique in that it is decentralized like a public blockchain while also making it possible to restrict the visibility of the information on the network like a private blockchain. It offers the benefits of both public and private networks, One can have a private permission-based system as well as a public permission-less system in a single network. Therefore with hybrid blockchain, we can make the ledger accessible to the public, with a private blockchain in the background that can control consensus and access to the changes in the ledger. The hybrid system of blockchain is flexible so that nodes can easily join a private blockchain. A transaction in a private network of a hybrid blockchain is usually verified within that network whereas transactions in the public network could be verified either in public network or in private network. The public blockchains increase the hashing complexity and rely on more nodes for verification of transactions. This event enhances the security and transparency of the blockchain network but increases load and decreases the performance.

1.2 Problem Statement

Stakeholders of the healthcare industry are operating in the technological stone age. Insurance providers rely on outdated medical data, clinics communicate ineffi-

ciently, and in turn, patients receive inadequate service. As other industries adopt technologies to make practices more efficient, healthcare has been forgotten and effectively left in the dust.[17]

Before we identify the problems existing with the status quo of the industry, let's first explain what EHRs[18] are and how they are currently being used. EHRs are a systemized collection of patient health data in a digitized form. EHRs include a range of patient information in the form of demographics, medical history, immunization records, and billing information. As we'll explore next, the type of data these records hold aren't where the problems exist but instead in how this data is being managed and stored.

The infrastructure that the EHR system was built upon wasn't meant to sustain the multi-institutional vast network of medical data we witness today. This robustness and lack of cooperation in the industry make data synchronization between different medical facilities a frustratingly long process. Unfortunately for stakeholders, these systems operate in siloed environments, making updates and changes to medical records an inefficient endeavor. Patient privacy is another point of contention as the systems that contain medical data have been proven to have inadequate security protocols[6]. Along with the inefficiency and vulnerability of these systems, the industry lacks a clearly defined line on where the liability of ownership and management of patient data exist. In the event of there being an attack or loss of data, it's hard to identify who's at fault and the appropriate steps to reduce the risk of future system failure.

EHRs presently exist in a void of "innovative" digital medical data, and inadequate infrastructure from the early 2000s[16]. By utilizing blockchain technology there is an opportunity to overhaul the outdated framework of digital medical records and eliminate some of the pitfalls of the industry. Blockchain integration motions to im-

prove communication inefficiency, the lack of transparency between providers, medical record sharing/updating, inadequate security protocols, and privacy concerns.

An important aspect of EHR systems is it's HIE or in general data sharing aspects. With a number of EHR systems being deployed in various health institutes, with different level of terminologies, technical and functional capabilities which makes it to have no universally defined standards. Hence, we need a versatile database that doesn't impose a need of structured data and can easily store, share and manage structured as well as non structured data(like images, videos, JSON files).

Stakeholders in the healthcare sector are concerned about privacy of data as well as membership of the network. Hence consortium blockchain solutions are an ideal choice for implementing an healthcare application for two main features:

1. Data Permissioning through Access Control.

2. Selective Membership to pre-approved participants.

1.3 Hyperldger Fabric

HLF is an open-source enterprise-grade permissioned distributed ledger technology (DLT) framework for developing solutions and applications. HLF provides some key differentiation functions on other popular distributed ledgers or blockchain platforms, such as plug-in consensus, smart contracts are written in a general programming language, thereby achieving innovation, adaptability, and optimization. Its highly modular and versatile architecture satisfies a broad range of industry use cases like banking, healthcare, supply chain management, finance, food industry, etc. [10]

The Fabric platform is also permissioned, that means, participants are known to each other and while they may not fully trust one another, a network can be operated under a governance model that is built off of what trust does exist between participants, such as a legal agreement or framework for handling disputes.

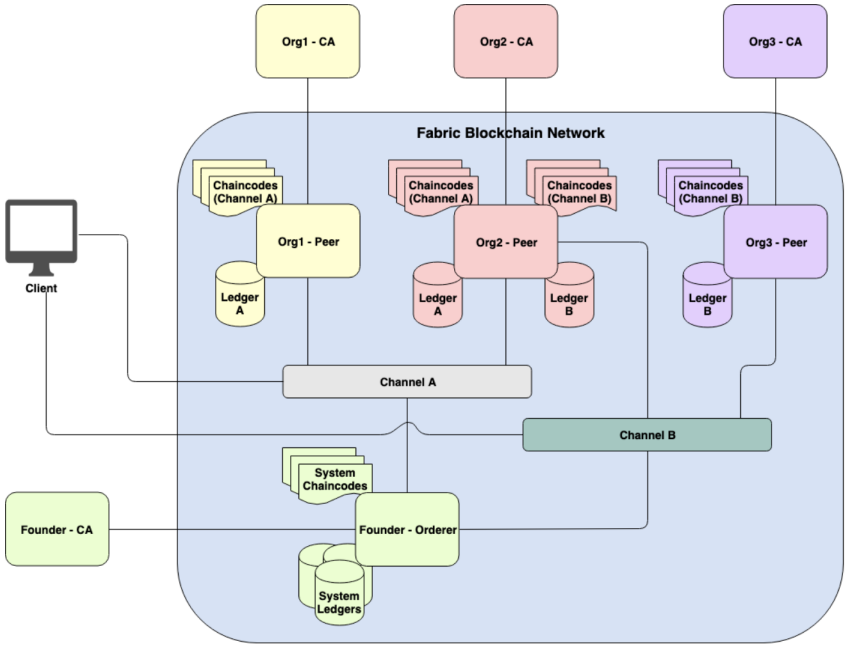


Figure 1.1: Hyperledger Fabric Architecture[2]

As HLF is permission-based and leverages consensus protocol, it does not require a native cryptocurrency to incent costly mining or to fuel smart contract execution. Avoidance of cryptocurrency mining reduces some significant risk/attack vectors, like 51% attacks where a network could be controlled by nodes if they have more than half of computing power. Also, it removes the significant energy cost associated with mining and keeps the overall operational roughly the same as any other distributed system.

Components of HLF :

- **Certification Authority:** A certification authority(CA) distributes certifi-

cates to all participants of the network. These certificates are digitally signed by the CA. HLF provides a built-in CA component called Fabric CA. It manages the digital identities of participants in the form of X.509 certificates. Each organization hosts a separate node that acts as a certification authority.

- **Organizations:** An organization is a managed group of members of the blockchain network. Every organization manages its members using a Membership Service Provider(MSP). An organization consists of peers, users, and clients.

- **Membership Service Provider(MSP):** An MSP identifies which Root CAs and Intermediate CAs are trusted to define the members of a trusted domain. An MSP can also identify specific roles an actor might play(eg. members or admins). An MSP can be implemented at an organization level in the blockchain network.

- **Peers:** These are nodes in the blockchain network that commit transactions as well as maintain the state and copy of the ledger. Peers can also act as endorsing peers(also called endorsers). Every chaincode specifies an endorsement policy that may refer to a set of endorsing peers that are used for validating transactions.

- **Clients:** These nodes submit transaction invocation to endorsers and broadcast transaction proposals to the ordering service. These nodes are usually controlled by end-users through an application. They must be connected with a peer of its choice.

- **Channel:** In HLF, there can be multiple ledgers shared between network members(organizations) to conduct private and confidential transactions. These ledgers or private subnet of communication are called channels. A channel is

defined by members(organizations), anchor peers of authenticated members, the shared ledger, chaincode application, and the ordering service node. Every transaction, either adding an asset or querying the ledger, is executed on a channel, where each party must be authenticated and authorized to transact on that channel.

- **Chaincode:** A smart contract is called a chaincode in an HLF network. It's a self-executing logic that encodes the rules for specific types of network transactions. Chaincode needs to be installed by all participating peers on a channel. Authorized peers can invoke chaincode through client-side applications. Chaincode transactions are appended to the shared ledger and modify world state, provided they are validated. HLF provides multiple programming languages for chaincode development.

- **Ordering Service:** The consensus algorithm in a public blockchain is probabilistic which guarantees ledger consistency, but is still vulnerable to divergent ledgers(sometimes called forks). HLF relies on deterministic consensus algorithms that avoid forks. For this solution, the ordering service is implemented using Apache Kafka and RAFT.

  Kafka is a Crash Fault-tolerant implementation of a messaging queue that uses a 'leader and follower' node configuration. Transactions are replicated from the leader node to the follower nodes. If a leader goes down, one of the followers goes on to become the leader, ensuring fault tolerance.

  RAFT follows a "leader and follower" model, where a leader node is elected (per channel) and its decisions are replicated by the followers. As RAFT ordering services does not require any additional services(like zookeeper) to maintain the cluster, it is easier to set up and manage than Kafka-based ordering services.

One more benefit of having RAFT is its dynamic leader design, where if leader node is down, rest of the cluster automatically chooses a new leader rather than waiting for leader to restart.

1.3.1 Hyperledger Fabric Transaction Flow

Figure 1.2 depicts the basic Transaction flow in HLF.

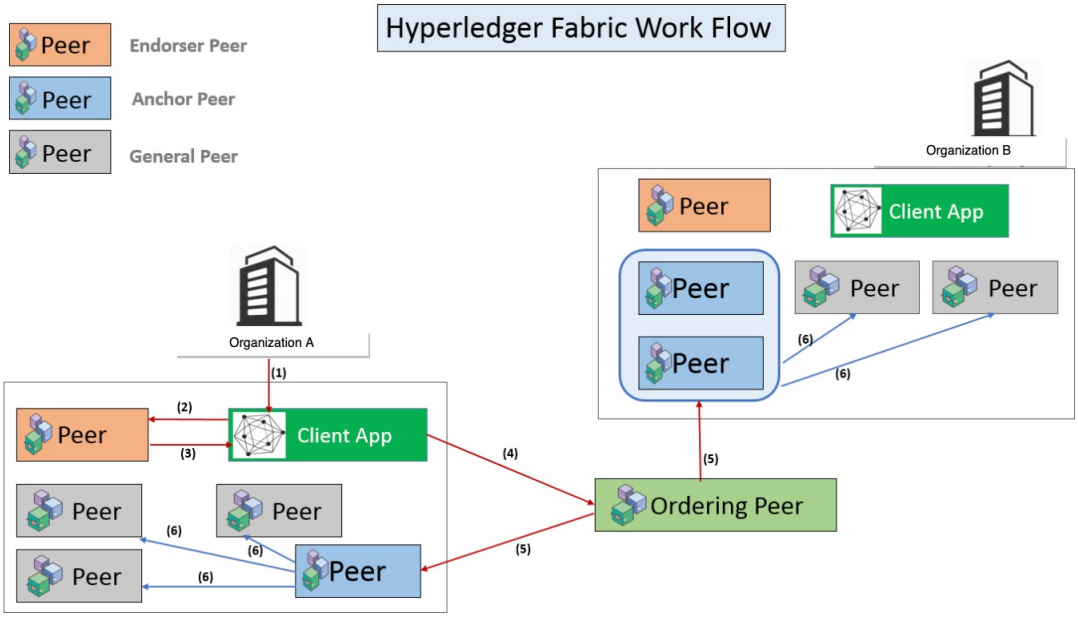


Figure 1.2: Hyperledger Fabric Transaction Flow[19]

1. An end-user in the member Organization invokes a transaction request on a channel through the client-side application that interfaces with an authorized peer.

2. The client application broadcasts the transaction invocation request to the endorser peer.

3. Endorser peer checks the Certificate details of the member and transmits the transaction request to others to validate the transaction. Then it executes the Chaincode (ie. Smart Contract) and returns the endorsement responses to the Client. Endorser peer sends transaction approval or rejection as part of the endorsement response.

4. If the transaction is approved, then the client sends it to the orderer peer for this to be properly ordered and be included in a block.

5. Orderer node properly arranges the transaction and include it into a block. Orderer node also forwards the block to anchor nodes(peers) of participating member organizations of the HLF network which are authorized in the provided channel.

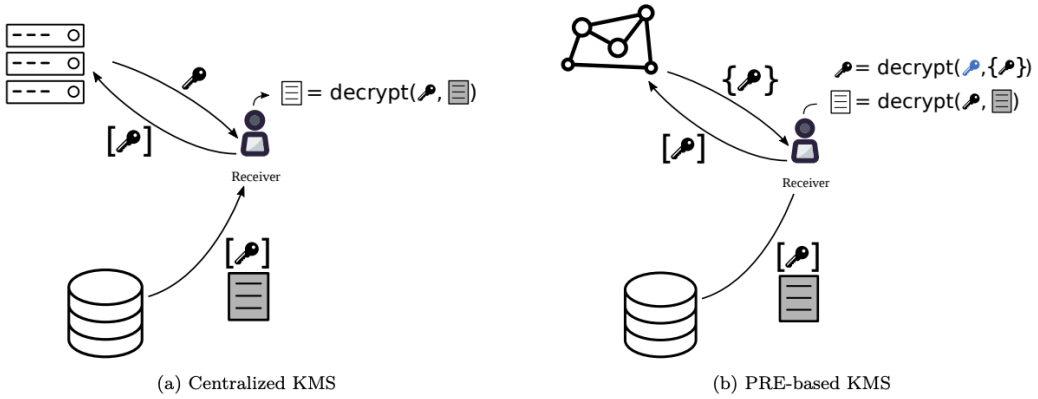
6. Once anchor peer receives the new block from the orderer node, they broadcast it within their own organization to sync ledgers on all peers to the latest block.

1.4 Nucypher

NuCypher[15] is a decentralized, blockchain based key management system (KMS), encryption, and access control service that can be either implemented on a private network or on the ethereum network(Currently development is going on in goerli test-net). It enables data sharing between users/nodes using proxy re-encryption over the network.

A centralized KMS is a system that controls the generation, distribution, and management of cryptographic keys for a broad range of applications. A KMS manages the entire lifecycle of cryptographic keys and manages to store them as well. However, due to being a centralized system, KMS as a service needs to rely on trust in a service

provider. Other than that, operation complexity increases significantly for a large number of organizations.



(a) Centralized KMS (b) PRE-based KMS

Figure 1.3: Centralised KMS vs Proxy Re-encrytion KMS

Nucypher uses a decentralized network to delegate encryption/decryption rights. By delegating rights among multiple nodes ensure reliability, availability, and correctness by having no single point of failure. By using proxy re-encryption, Nucypher makes sure that unencrypted symmetric key and re-encryption keys are not accessible at once, keeping data protected.

Decentralized KMS helps to conditionally grant and revoke access to sensitive data to arbitrary numbers of recipients. Nucypher does that through policies that can be either time based or condition-based.

1.4.1 Proxy Re-encryption(PRE)

Public-key encryption(PKE) is a type of asymmetric cryptography where each sender and receiver needs a pair of keys: public and private keys. The advantage of PKE is that two parties can exchange information without any required common secrets. In such systems, a party can encrypt a message using the receiver's public key but requires the receiver's private key to decrypt that message.

PRE is a special type of PKE, where a proxy entity(third party) is allowed to alter a ciphertext from one party(sender) so that another party can decrypt it, without knowing the message. In a typical scenario, there are mainly three actors, data owner, proxy and delegatee. Suppose, Alice, the data owner, has key pair pk_A/sk_A. Alice wants to share some message m, with Bob, who has a pair of keys pk_B/sk_B. Alice encrypts the m using PRE function $rk_{A\to B}$ and her public key pk_A, resulting in ciphertext c_A.

$$rk_{A\to B} = rekey(sk_A, pk_B)$$

Importantly, in single-use, uni-directional PRE schemes, this re-encryption function is one way, and $rk_{A\to B}$ cannot be decomposed into its component parts. So, PRE transforms the ciphertext from c_A to c_B so that Bob can decrypt it.

$$c_B = reencrypt(rk_{A\to B}, c_A)$$

Some benefits of PRE systems are Alice can leave the system after encryption. Other than that, delegation helps secure sharing through multiple organizations without exposing encryption mechanism to third party.

1.4.2 Advantages of Nucypher

Nucypher has numerous benefits, some due to it's decentralization and others due to flexible sharing policies.

1. **Unidirectional, single-hop, non-interactive:** Nucypher is based on a non-interactive algorithm. "Interactive" means a re-encryption key is computed using both private keys, senders and receivers.

$$re_{ab} = rekey(sk_a, sk_b)$$

. While non-interactive algorithms do not need private key of delegatee, but only requires the public key.

$$re_{ab} = rekey(sk_a, pk_b)$$

This way non-interactive algorithms ensures the safety of receivers.

Nucphyer is also unidirectional, which means it is impossible to compute re_{ba} from just re_{ab}. This way, Bob never needs to reveal it's private key to the system.

PRE algorithms can also be single-hop or multi-hop. Multi-hop means that multiple re-encryption keys can be sequentially used to convert ciphertexts without the participation of any intermediate re-encryption keys. Suppose, re_{ba} and re_{bc} are two re-encryption which are generated by participants a,b and c. Then ciphertext c_a could be converted to ciphertext c_c without any participation from b.

$$c_c = reencrypt(re_{bc}, reencrypt(re_{ab}, c_a))$$

This is not possible in single-hop algorithms.

2. **Versatile Data Sharing Policies** Nucypher can be used either on a private network or on an ethereum based network. Ethereum based Nucypher allows smart-contract based data sharing policies. While we don't trust arbitrary miners in the system with encrypted data or re-encryption keys, we still trust them to create policies that control the duration of encryption.

   Policies can be time-based: re-encryption is allowed only during a specific time interval and re-encryption key should be removed once the time is up. More complex policies like based on the success of transaction or based on the result of a condition, data could be shared.

3. **Splitting trust across re-encryption nodes**

   Due to it's decentralized nature, you can mention in policy to split trust among multiple re-encryption nodes and specify a number of minimum nodes required to provide keys for decryption. More reliability and availability can be ensured this way as splitting node means more up-time.

1.5 IPFS

IPFS[12] is a protocol and peer-to-peer distributed network to connect all devices with the same file system. IPFS uses content-addressing to uniquely identify files in a global file system. In other words, IPFS provides a high throughput content-addressed distributed file system forming a generalized Merkle Directed acyclic graph(DAG).

IPFS uses the hash of files to uniquely identify them, so if somebody uploads two files with the same hash value, IPFS pin them to the server and creates only one entry in the content-addressed storage block. Due to this feature, IPFS has no single point of failure. IPFS has no access control and files over IPFS can be directly accessed by respective hash values. This makes it easier to integrate with different blockchains as an off-chain storage solution where blockchain act as an access control system.

1.6 Merkle Tree

In cryptography and computer science, a hash tree or Merkle tree[8] is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the hash of the labels of its child nodes. Each data value at leaf nodes is converted to corresponding hash values, while non-leaf nodes are calculated with the summation of hash values of their child nodes and then re-hashing them[5].

Merkle tree is constructed from the bottom up, and the head of the tree is known as Merkle root or hash root. The advantage of Merkel tree is that it can use any hashing algorithm or a combination of hashing algorithms. This process can be used on any kind of data and any amount of data. The Merkel root summarizes all of the

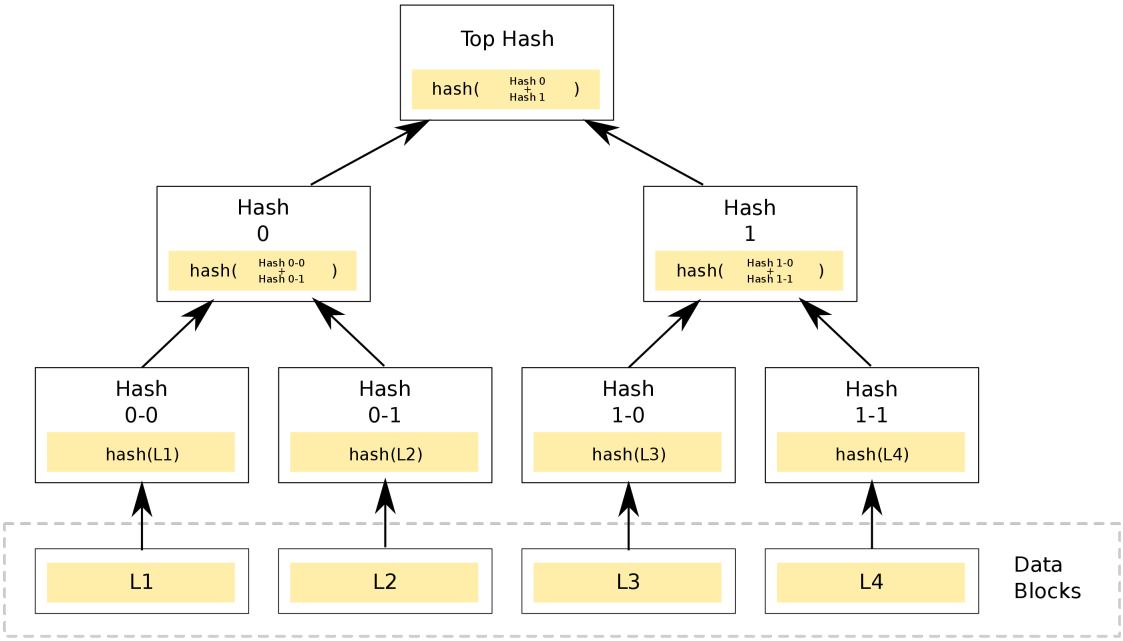


Figure 1.4: Merkle Tree[8]

data in the tree and can be used for data verification. Any change in data blocks could entirely change the Merkel root value. Merkel tree allows for a quick and simple test whether a specific data has tampered or not.

1.7 Why a Decentralised Healthcare System ?

EHRs were never meant to manage multi-institutional, lifetime medical records. With a sudden boom in distributed storage and lack of rules, providers rushed to create their own siloed systems of EHR management. As a result, when patients

change services, they will spread healthcare data among multiple providers, HIE and insurance companies. In doing so, they lose easy access over data and sharing rights. HIPAA privacy act allows providers to take up to 60 days to respond to a request by a patient for updating or deleting a record that was erroneously added[20]. In addition, because the patient's medical records are scattered in different organizations and rarely can be fully viewed, the maintenance of medical records may face great challenges. This jeopardizes the health care services provided to the patient. If the hospital is not part of the HIE, data sharing between different providers and the hospital system will be very difficult. The lack of coordinated data management means poor medical services and outdated records management.[11]

Patients may face significant hurdles in personal healthcare data retrieval and sharing them among multiple providers due to economic incentives that promote "health information blocking." A recent ONC report[21] details several instances where developers of healthcare applications interfered with the flow of data and then charging exorbitant prices for data exchange.

The above obstacles occur because the system is not patient-centric and data management is not directly provided to patients. When designing a new system, we must give priority to patient agents. A patient-centric approach can help reduce fragmented data, improve record sharing performance, and reduce the economic cost of EHR management. However, such systems should also recognize that patients cannot be trusted with their own data, and they may tamper with it due to economic benefits.

Healthcare and EHR data is very sensitive and is protected by both Federal and State laws. With underlying Health Insurance Portability and Accountability Act(HIPAA) privacy guidelines, Patients are afforded specific rights regarding the

sharing of their information through an Health Information Organizations(HIO)[3], including but not limited to:

- The right to opt-out of having their individually identified information securely shared through the HIO.
- The right to change their mind regarding an opt-out decision.
- The right to opt-out of a particular healthcare provider sharing their health information through the HIO.
- The right to ask for a copy of their health information that is available through the HIO.
- The right to request that incorrect health information about them be amended.
- The right to request a list of individuals who have viewed their information through the HIO for a period of at least 3 years before the patient's request.
- The right to be notified of a breach at the HIO that affects the patient's individually identifiable health information.

Medical records are a significant part of healthcare research as well. The ONC's report[21] emphasizes that biomedical and public health researchers "require the ability to analyze information from many sources in order to identify public health risks, develop new treatments and cures, and enable precision medicine". Due to the recent increase in demand for healthcare data, we need a responsible data-sharing model that can ensure that patient data is successfully de-identified before sharing with research institutes.

In this work, we will explore a blockchain based patient-centric decentralized solution for managing and sharing EHRs. This solution can share EHRs on time based

conditions and even share need-to-know basis. We build a system with HLF for Identity and Access management. Being a permissioned blockchain, Fabric will only allow authorized patients, providers, insurers and researchers. For storage, we will use IPFS, which each organization can have a node or different organizations can share node. Although, IPFS is a peer-to-peer system, access is controlled by HLF. Due to public nature of IPFS, we are using Nucypher for encryption and policy control. Time based policies help us to control access to shared materials and easily be revoked or extended based on requirement. Although HLF provides immutable ledger but IPFS on itself doesn't ensure tamper proof storage. For example, A patient for it's benefit, can tamper records before presenting it for insurance claims. To address this problem, we used merkle root, which would be created with every document and merkle root will be shared on HLF. This way, a record could be checked against merkle root with merkle proof. Merkle tree also enables partial data sharing for some text-based EHRs, as each shared part of record could be verified using merkle proof.

With HLF, Nucypher and IPFS, we build a system that can integrate with almost every HIE. MedFabric4Me implementation features :

1. Creating, sharing, deleting and updating of EHR records among Patients, Insurers and Providers. Patients can share records on need-to-know basis while updating a record needs providers consent.

2. Sharing of EHR records outside network or among researchers where records can be verified but can't be updated or created. Particiapnt may be part of network or not.

MedFabric4Me addresses major issues currently faced in healthcare industries:

- **Slow access and sharing of EHR** As Insurers and Providers own most of the EHR records, it is extremely difficult to share data by patient to different

parties. Presently, HIE is the best solution to tackle these problems, but even HIE is connected directly to different providers and insurers, not patients. This makes accessing, sharing and even correcting EHRs extremely tedious and long process. MedFabric4Me tackle this problem as a patient centric system, owning all of their own data, patients can share their records within minutes and similarly can revoke access within minutes.

- **Fragmented data** There are no central repositories for all EHRs, and thus data moves from one providers silo to another. HIE currently stores data at state level but doesn't provide a scalable solution to national level. This problem quickly escalate when patients move further away, In doing so they lose easy access to past data, as the provider, not the patient, generally retains primary stewardship.

- **System Interoperability** Due to no standard definition or format for EHRs, each organization has it's own structure to store and share EHRs. Quality of data decreases when these records are shared among different organization to no standardisation. MedFabric4Me provides a common platform and thus a standard for sharing and storing EHRs.

When designing any solutions related to EHR, we should keep in mind these basic rights provided to patients.

1.8 Hypothesis

A healthcare system that challenges HIE should have performance and Scalability of HIE. Our system should have following properties:

- **Privacy:** MedFabric4Me should be able to comply with HIPAA privacy rules mentioned in Section 1.7.

- **Selective Data Sharing:** MedFabric4Me should be able to share partial data to show compatibility for de-identification data.
- **Scalability:** MedFabric4Me should be scalable enough to compare with currently present state of art HIE systems.

We hypothesize that MedFabric4Me should behave in following ways :

1. **Kafka vs RAFT:** RAFT is an ordering service that necessarily requires TLS certificates for processing transactions while Kafka don't. TLS certificate requirement should increase latency and decrease average throughput, hence Kafka should perform a little bit better than RAFT.
2. **HLF Performance in Throughput:** Distributed systems tend to decrease with increase in decentralization. So with increase in number of peers, throughput should decrease for both Kafka and RAFT. But even for high number of peers, MedFabric4Me should be able to provide higher throughput than currently required state of art solution.
3. **Adding Asset vs Query in HLF:** HLF requires to satisfy endorsement policies for adding an asset while querying doesn't need that. Therefore, Querying should have higher throughput than adding assets to ledger for same number of peers and same ordering service. This is a significant measure because HIE's have higher query requests than adding new records.
4. **IPFS downloading and uploading speed:** IPFS provides a hash value based on content of file, if file can be processed. On the other hand, contents of encrypted file can't be processed and thus generates hash based on file name. Thus, uploading time for encrypted files should be significantly less than non-encrypted files.

Downloading of file should not depend on type of file and should have a linearly proportional relationship between time and size of file.

5. **Nucypher Encryption and Decryption:** Nucypher is a PRE scheme and should provide a linearly proportional relationship between time required for encryption/decryption and size of file.

We implement MedFabric4Me in minimal-viable-product way. The tests involves changing the HLF parameters, Nucypher encryption- decryption metrics and IPFS benchmarking against different record sizes.

1.9 Organization of Thesis

The content of thesis is organized as follows :

1. **Chapter 2:** Focuses on Background and Related Work in healthcare industry
2. **Chapter 3:** Describes the design and implementation of MedFabric4Me
3. **Chapter 4:** contains the Experimental Details. It contains information for experimental setup, data set, evaluation criteria and performance of the system.
4. **Chapter 5:** concludes the thesis and describes the direction for future work.

Chapter 2

RELATED WORK AND RESEARCHES

2.1 Health Information Exchange - healthcurrent.org

Electronic HIE allows health care providers and doctors to appropriately access and securely share a patient's vital medical information electronically—improving the speed, quality, safety and cost of patient care.

The major advantage of HIE is for providers rather than patients. Providers are the primary users of HIE and use it for the following benefits: [7] :

- HIE provides access to and retrieval of healthcare data to provide safer, more timely and efficient patient care.
- Reduce expenses related to the manual handling of records. This reduces the cost of maintenance and manual backup.
- HIE provides a vehicle for improving quality and safety of patient care by reducing medication and medical errors.

Although HIE has some benefits for patients as well, being aggregating all medical data at one place but it never meant to scale and support patients.

Some of the major challenges that HIE faces in general :

- **Privacy and Security Concerns:** Health records are vulnerable during the gathering, sharing and transmitting.
- **Inconsistent Policies and Laws:** Policies and laws differ between federal, state, regions and even hospitals and need to be standardized.

- **Slow response to privacy laws:** A patient could appeal to update or delete a record from HIE, but a HIE or provider could take up to 60 days to execute that. Similarly, exchanging or updating information from one HIE to another could take months to execute.

- **Limited visibility of data for patients:** HIE is not bound to share data with patients until patient appeals. This causes limited visibility of data for patients and harder to share information out of network.

Our solution follows HIPAA guidelines closely and rather than providing a replacement for HIE, we present MedFabric4Me as a blockchain based innovative solution to curb HIE's challenges for data privacy, sharing and integrity.

Healthcurrent is the HIE that helps partners transform care by bringing together communities and information across Arizona. Healthcurrent manages more than 674 care providers serving 9.6 million patients and performs more than 95 million transactions per year.

2.2 MedRec Prototype in EHR and Medical Research

MedRec[11] is an ethereum based record management system for EHRs. MedRec leverages blockchain to manage user authentication, the confidentiality of records, accountability and data sharing. MedRec provides the patient with a complete history of their medical records, immutable log and easy access to their medical information across different providers. Rather than creating a new and ideal solution, MedRec easily integrates with existing local data storage solutions, facilitating interoperability and making system convenient and adaptable.

Ethereum requires mining mechanisms to sustain a distributed ledger, where different nodes compete for proof of work. Mining is a time consuming, and expensive

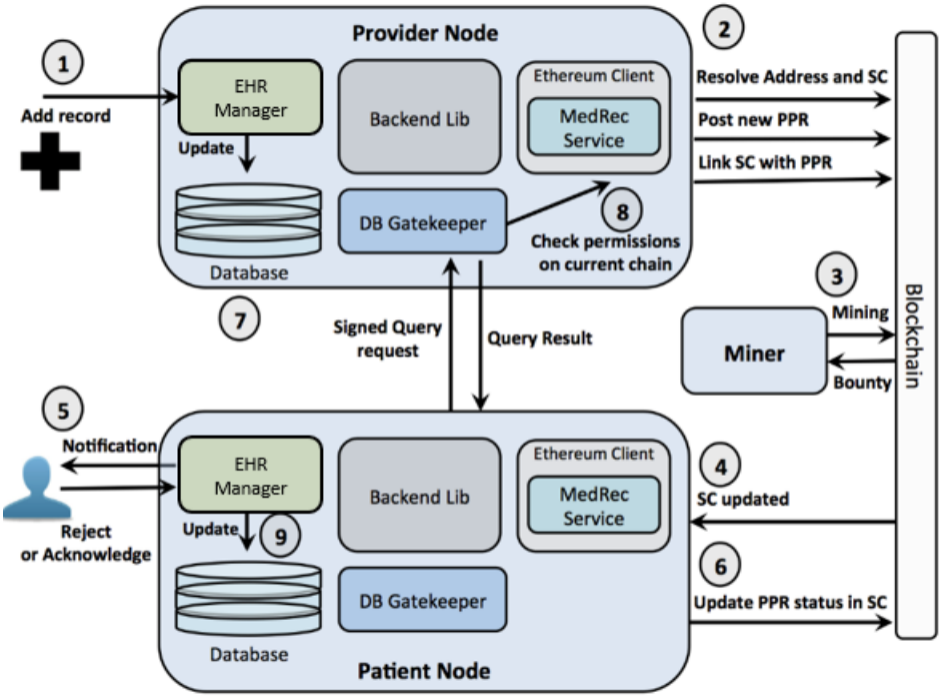


Figure 2.1: MedRec System Orchestration[11]

process to execute. Other than this, MedRec only integrates with provider's current storage solution rather than presenting patient centric, patient owned data economy. This project is supported by a grant from the Robert Wood Johnson Foundation, with additional support from the MIT Media Lab.

2.3 Ancile: Privacy-Preserving Framework for Access Control and Interoperability of EHRs

Ancile[13] is another blockchain based system using the private ethereum platform, which applies a technique that is similar to ours for storing EHRs, adopting on-chain and off-chain concept. Ancile uses distributed proxies for re-encryption, called blinding re-encryption, by splitting the ciphertext for re-encryption between multiple nodes.

Using smart contracts, Ancile maintains cryptographic hashes of stored records and query links, confirming the integrity of EHR Databases. Patients can also view and control who has permissions for their private information by using a smart contract to manage access control. Moreover, patients may give transfer permissions to other nodes. This is possible through the use of identity-checking, to confirm who may access records, and PRE, to avoid having to re-encrypt the record for each transfer.

Although Ancile uses off-chain data, it's data sharing policies and data writing accesses are controlled by ethereum.Primary limitations of public blockchain are that collusion of 51% of mining nodes on the system could result in the rewriting of the chain structure.

2.4 Secure and Trustable Electronic Medical Records Sharing using Blockchain

In this concept, Dubovitskaya[14] propose a HLF in cloud based system. This system incorporates an encryption system as well, where patient encrypt each piece of their data with a different symmetric keys. Dubovitskaya shares different scenarios in which their application could be used. They also presented an architecture of the framework for the specific needs in case of radiation oncology data sharing and implemented a prototype that ensures privacy,security, availability, and fine-grained access control over highly sensitive patients' data.

But due to multiple symmetric key encryptions of different data, it causes heavy burden on the system. Also, need to implement a key management system on patient side.

2.5 Comparison of MedFabric4Me with State of Art Systems

In this chapter we have seen multiple systems utilising blockchain technology in Healthcare sector, specially in storing ans sharing EHR data. There are some similarity between these systems with MedFabric4Me:

- Data storage is off-chain, only hash values are shared either for logging purpose or sharing purpose.
- Blockchain technology is used for privacy and verification of records for it's immutability.

Although there are some similarities between state of art systems and MedFabric4Me, there are some features that MedFabric4Me provides which were lacking in above systems:

- MedFabric4Me is patient-centric and enables personal data economy, that means patients own their data and are able to create, share, delete and update their own data records.
- MedFabric4Me provides a PRE along with distributed file system that can store structured and unstructured data with encryption. Having a distributed layer of PRE not only adds more security, it prevents single point of failure as well.

We will cover more aspects of MedFabric4Me along with architecture and design in next section.

Chapter 3

METHODOLOGY

3.1 Hyperledger Fabric

In HLF, there are several key components that play pivotal roles in the system. In addition, it provides three phases of consensus to validate transactions before uploading them to the ledger.

Key components of HLF that help our system:

- **Member Service Provider:** MSP abstracts away all cryptographic mechanisms and protocols behind issuing and validating certificates, and user authentication. An MSP may define their own notion of identity and the rules by which those identities are governed (identity validation) and authenticated (signature generation and verification).

  An MSP can identify specific roles an actor might play within its organization (e.g admin or a patient or doctor), and sets the access privileges in the context of network and channel.

  Our system extensively uses MSP services to register Patients, Providers, Insurers, and Researchers, and assign access privileges based on their roles and organizations. For e.g, a doctor may write and read a patient record on permission but a researcher can't write a medical record, only read.

- **Endorsement policies:** Endorsement policies define the condition or agreement for a transaction to be valid. Endorsement policy can widely vary based on the requirements, like a peer from an organization can endorse a complete

transaction or all organizations may need to endorse for valid transactions. when the ordering service sends the transaction to the committing peers, they will each individually check whether the endorsements in the transaction fulfill the endorsement policy. If this is not the case, the transaction is invalidated and it will have no effect on world state.

Endorsement policies helps us make sure that a transaction is verified by trusted parties and make sure that no malicious actor is trying to change world state ledger.

- **Channels:** A channel is a private subnet of communication between two or more specific network members. In our application, we create different channels to segregate data according to it's usage. For e.g, Merkle root of a document should be shared with all parties involved and should be shared in a channel in which, all parties are enrolled. But sharing of data among patients and providers should be done in a different channel, with different endorsement policies and user roles.

Other than above stated features, we used chaincodes to execute to facilitate, verify and enforce negotiation and agreement between users. Chaincode has carefully written programming functions in it, that helps users to read and update ledgers as per permissions. Different channels deploy different chaincodes as per needs.

3.2 Tamper Proof Off-Chain Storage

Off-chain data is any non-transactional data that is too large to be stored in the blockchain efficiently, or, requires the ability to be changed or deleted. In practice, medical data is usually too big to handle directly in a ledger; therefore, data is kept in IPFS, and only the address is recorded in the ledger.

IPFS is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS is particularly useful for our use case due to following properties :

- **Easy sharing of data:** Sharing protocol over IPFS is really simple, just share the hash value provided by IPFS when saving data or file. This protocol allows easy sharing among different nodes and peers.
- **No restricted file type:** IPFS is unstructured data storage. You can store any kind of file or data on IPFS. This is important for MedFabric4Me as, Medical data is widely sparse and can have multiple formats.
- **Tamper proof:** IPFS creates hash value not only depending upon the file name but by the content of file as well. It is extremely difficult to change content of file and yet receive the same hash value. This way, data can not be overwritten in IPFS and ensures data integrity.

Moving storage off-chain alleviate some of the concerns of our application, like :

- The data no longer needs to be hosted by all nodes but only by the nodes that have access to data. For e.g : As a patient centric system, a patient only needs to store data on their IPFS node related to their EHR. Similarly, Doctor can only pin data that he has permission to. This way, Off-chain data storage reduces the storage requirements that each node is expected to store.
- Off-chain computation reduces to high extent redundant computations required to achieve consensus. Heavily reduces the processing and querying of on-chain data, as only a fingerprint of the data is stored on-chain.
- A peril of off-chain data storage is data availability is no longer guaranteed since the data is not part of the blockchain, but IPFS is a peer-to-peer decentralised

system ensuring that any important data is not only highly available but is distributed among different physical machines making it resilient to failure.

3.3 Cryptographic Scheme

Before patient data is uploaded to the IPFS with the patient's consent, the data is encrypted using nucypher network. Nucypher network encrypt data with patient's private key and public key. Self-encrypting data ensures that other patient nobody can access the data, even if it is pinned over IPFS network.

Now whenever a patient wants to share data over the network, it can re-encrypt the data using symmetric key system. Then the symmetric key is asymmetrically encrypted using the patient's public key and attached to the encrypted data. This hybrid encryption makes the procedure efficient in terms of both speed and convenience because the encryption of large data can be done faster by symmetric-key than asymmetric-key, while the latter is more convenient in the encryption of small-size cryptographic key.

Nucypher provides policies to provide reading rights. This means, that if you want to read an encrypted data, you should have a policy with patient/sharing party. Versatile policies states how long data can be decrypted and who can decrypt it. Once the policy is decided it shared over the IPFS, along encrypted data file.

The encrypted symmetric key at the data should be transformed, so that it can be decrypted by the receiver's private key. To do this, Nucypher use a PRE scheme (Figure 2) in which the patient generates the PRE key by mathematically combining their private key and the receiver's public key. After receiving the newly made re-encryption key, the proxy re-encrypts the symmetric key for the receiver. In that process, the symmetric key is not disclosed to the proxy. Otherwise, the proxy must send the data to the patient to make it encrypted using the receiver's public key.

Nucypher's PRE scheme has many benefits that helps us in our application :

- **Versatile data sharing:** Nucypher's policy creation helps us share data with versatile policies. Time based conditional policies helps patient to revoke access after a certain period in time, arbitrary smart contract based conditions could be used for more complex scenerios as well.
- **No need to encrypt multiple times:** Due to different than encryption policy system, there is no need to encrypt data with symmetric key every time. Once encrypted data can be re-encrypted using different asymmetric keys using PRE, allowing patients to encrypt only once but share with multiple parties using policies.
- **High availability:** Nucypher is a ethereum based decentralised PRE system which can split trust among multiple nodes. This allows sharing of data highly available, even in case if a node or two goes down, it doesn't effect availability as key is splitted among multiple nodes.

3.4 System Design

In our system, there are four main actors/organizations :

- Patients: These are main data owners, as a patient centric application, all the personal data and EHR data will be held by patient itself.
- Providers: Providers are another important actors in our application. They can create EHR data for patient. They can also update already created records and delete on request. Patient can give read and write access to providers on need to know basis.

- Insurers: Insurers are majorly insurance companies that plays a role of accessing data for claims.

- Researchers: Researchers are another type of actors that can access patient's data with consent. This data will be deidentified for privacy concerns. This actor can ask for data from patient's or patient's can share some data for research in advance without even knowing researcher.

We utilize HLF's chaincodes (Smart contracts) to intelligently create, update, delete, share and track EHR. It builds into the blockchain a Turing-complete instruction set to allow smart-contract programming and a storage capability to accommodate on-chain state. We regard the flexibility of its programming language as an important property in the context of EHR management. This property can enable advanced functionality to be coded into our proposed system, adapting to comply with differences in regulation and changes in stakeholders needs.

HLF supports multiple channels which is particularly useful to segregate data based on roles and requirements. Each channels may have multiple smart contracts and each channel manages it's own ledgers. This way no two channels can interact with each other keeping actors roles and data well defined.

In our application there are four channels :

- **Common Channel:** Common channel is used to share Merkle root of records. As soon as a doctor creates or update a record, it's merkle root should be shared on the common channel. Common channel comprises of all four actors, this way anyone with data can compare hash proofs with merkle root and compare whether shared data is tamper proof or not. Common channel also stores identity of actors along with other public credentials like Nucypher public key and ethereum addresses.

- **Patient-Provider channel:** This channel stores all information related to interaction between patient and Provider. This channel stores information related to sharing between patient and provider, creating/updating/deleting a new document, history of interactions and also saves data sharing policies.

- **Patient-Insurer-Provider channel:** Patients can raise claims here against insurance companies, which could be verified by providers. This way, insurance companies can have selective data from patient which provider suggests to share. This channel can also help in real time tracking of insurance claims.

- **Patient-Researcher channel:** This channel is intended to share records with researchers. This could be done in two ways,

  - Researcher provides it's public key for nucypher and then patient shares it with them.

  - Patient creates a public-private key from nucypher system and shares it on this channel. This way if any researcher is interested in data can access it.

  Although different in sharing methods, both methods controls sharing through nucypher policies and researchers can verify data integrity through merkle proof.

One important aspect of this system is the provision of endorsement policies. In the blockchain based system an endorsement policy decides whether a transaction is approved or not. For a smart contract an endorsement policy decides the number of approval signatures it should receive before a transaction is committed to the ledger. If a malicious actor tries to modify the ledger, trying to take it to an inconsistent state, the Read-Write set would create a conflict and the transaction won't make changes to the state of ledger.

The endorsement policies for smart contract deployed on all channels for our application are as follows:

- **Common Channel:** AND(PatientMSP.member, ProviderMSP.member, InsurerMSP.member)
- **Patient-Provider channel:** AND(PatientMSP.member, ProviderMSP.member)
- **Patient-Insurer-Provider channel:** AND(PatientMSP.member, ProviderMSP.member, InsurerMSP.member)
- **Patient-Researcher channel:** AND(PatientMSP.member)

3.4.1 Creating an EHR Record

MedFabric4Me supports structured, unstructured or binary data(Images, videos) as a valid EHR file. As files are received through IPFS, providers have no restrictions over format of data. Our application first modify the records with Merkle Tree implementation and then stores it in IPFS. In our application, EHR records can only be created by Providers but with the consent of patient. Providers can vary from hospitals, doctors and clinics. After having a visit, a patient should give access to a doctor to create a record. Once record is created, a patient can revoke the access from further visibility for provider. HLF records this transaction in ledger with provider ID, Patient ID and encrypted document's hash. It also stores Merkle root in common channel.

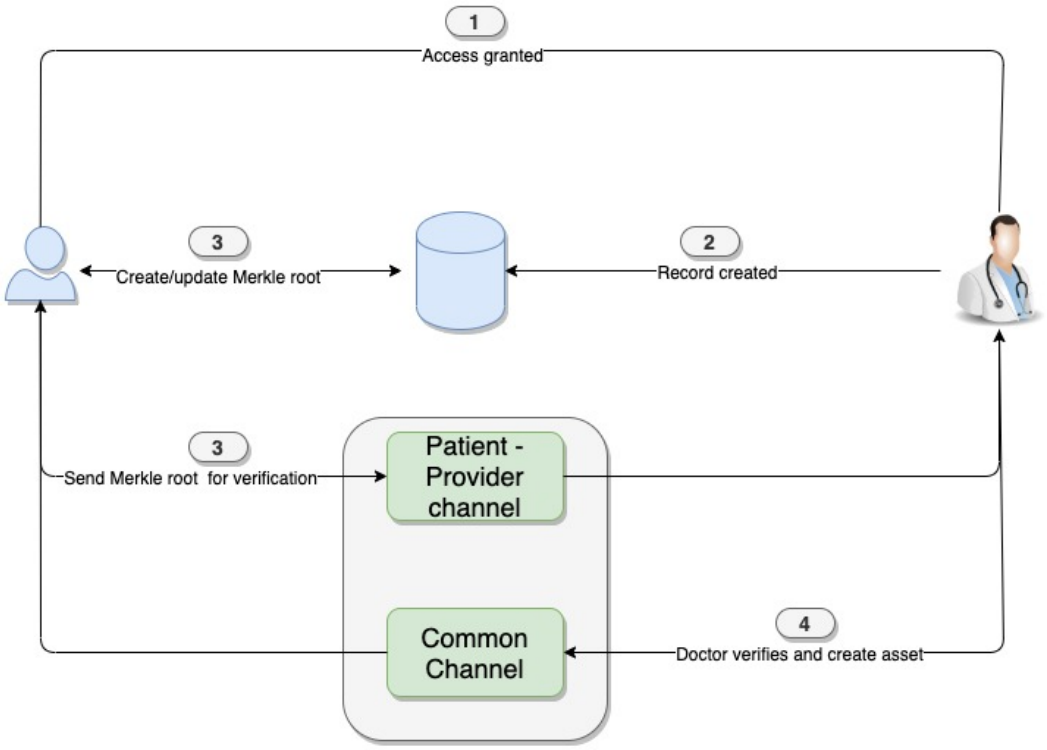


Figure 3.1: EHR Record Creation with Steps Numbered

The complete procedure looks like :

1. **Access by patient:** Patient provides access to provider to create an EHR for patient.

2. **EHR creation:** Provider creates the EHR document, encrypt the document, uploads to IPFS and update HLF with transaction.

3. **Merkle Proof:** Either an already existing tree get updated or new merkle tree is being created by patient in lieu of EHR, then Provider verifies the Merkle root and update it in common channel.

4. **Further Access:** Once the process has been completed for document creation, a patient verifies the record and either revoke, request an update on being wrong data entry or extend the access for further visits.

The smart contract used for creating a new document first verifies provider and Patient through MSP and then create a transaction proposal. Then this transaction proposal is either approved or discarded based on endorsement policy based consensus.

3.4.2 Updating an EHR Record

Just like creating an EHR record, updating is also only authorized by provider. A patient give access to provider for updating a record, which can be a complete record or can be a partial document. Once authorized a patient can make changes to document and doctor can verify it. HLF updates the asset and Merkle tree corresponding

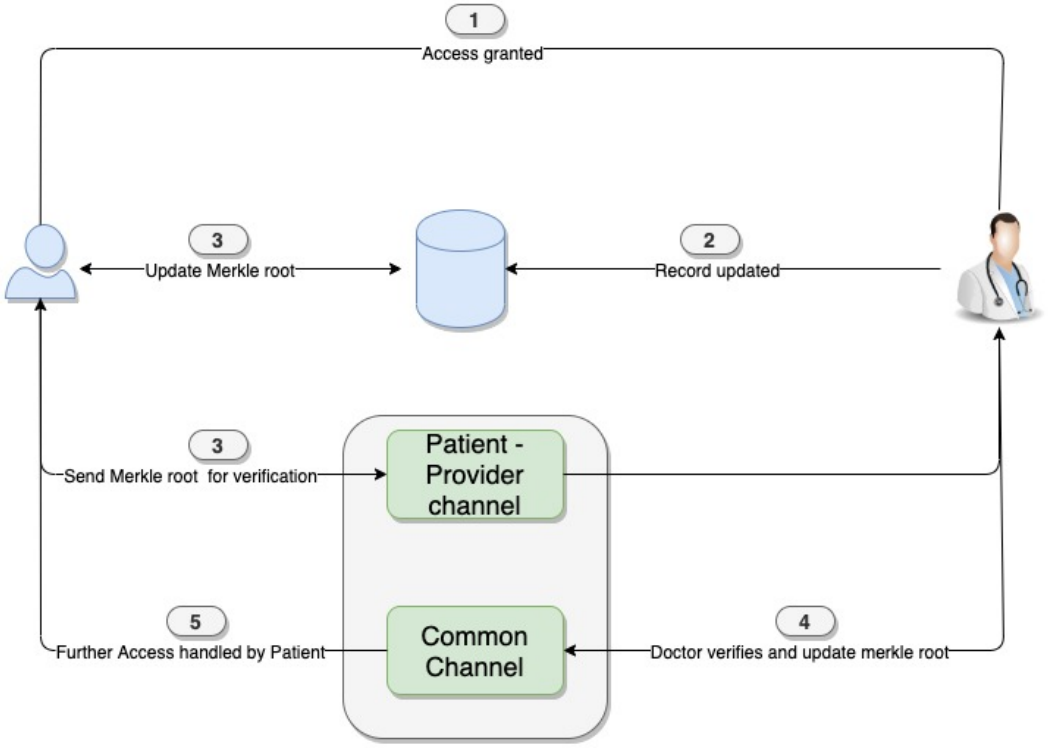


Figure 3.2: EHR Record Update with Steps Numbered

to that record.

The complete procedure looks like :

1. **Access by patient:** Patient provides access to provider to update an EHR.
2. **EHR update:** Provider updates the EHR document, encrypt the document and update HLF with transaction.
3. **Merkle Proof:** Patient creates new document with update and updates the Merkle Tree. After updating merkle tree, share the root with doctor.
4. **Verification:** Doctor verifies the merkle root with proof and and updates the merkle root asset accordingly.
5. **Further Access:** Once the process has been completed for document update, a patient verifies the record and either revoke, request an update on being wrong data entry or extend the access for further visits.

3.4.3 Deleting an EHR Record

Every EHR is created with a Provider ID and Patient can request provider to delete subsequent entries about an EHR and Merkle root asset from HLF accordingly.

The complete procedure looks like :

- **Request by patient:** Patient requests the provider to delete the asset from HLF.
- **EHR deletion:** Provider reviews the document, and if found legible to delete, deletes the EHR asset from HLF.
- **Updating Merkle Tree:** Once provider deletes the asset, patient updates Merkle root accordingly.

- **Verification:** Doctor verifies the merkle root with proof and and updates the merkle root asset accordingly.

Asset deletion from HLF doesn't mean that transactions are deleted as well. Transactions in HLF are immutable and if required, deleted records history could be traced on HLF.

3.4.4 Sharing an EHR Record

Sharing an EHR record can be done in two ways. EHR records varies from Structured data to unstructured data. EHR records could simply be a note from doctor and they can be thousands of images stacked together. For sharing all kinds of information, we have divided sharing into two parts :

- **Complete document sharing:** A patient can share full data record with provider, insurer or researcher, just by sharing merkle proof.
- **Sharing partial data:** A patient can share parts of a document, by sharing corresponding merkle proof of document as well as that partial record.

It is important to notice that all documents can't be partially shared. Our applications, shares partial data of only structured EHR records. Sharing data with different stakeholders comes at different risk levels. Insurers may want to see history of records while providers may want to look at large set of data. On the other hand, researcher only wants very specific information and may need to de-identified data for privacy. To achieve this we divided our application in different channels, applying different sharing policies of data among stakeholders. Different sharing policies with stakeholders:

- **Sharing with Providers:** Providers are registered members on Nucypher and has already been provided with a Public and Private key set. If a patient

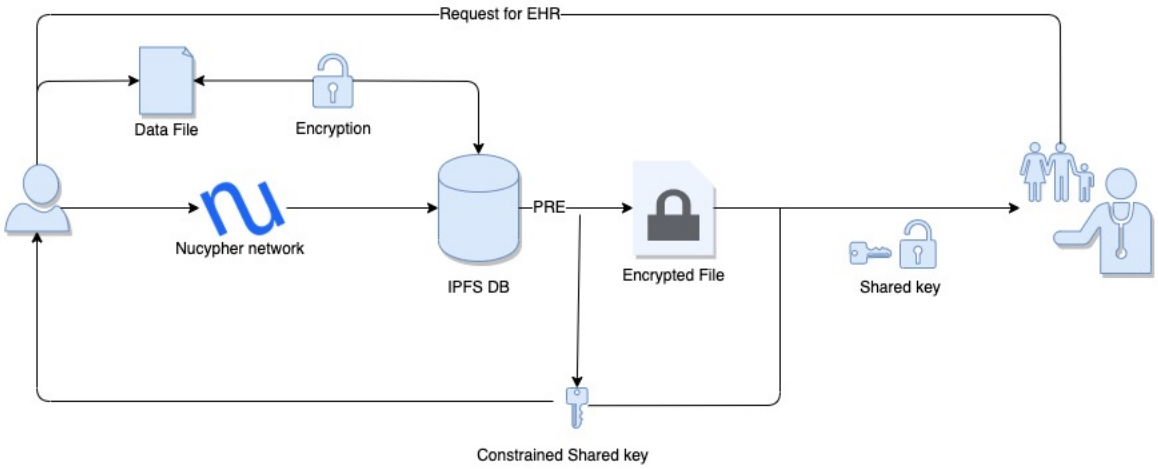


Figure 3.3: Basic Sharing Structure

wants to share data with provider, it can use this key set to create a policy and share data with doctor. Doctor may need to share data back with patient, to update or create a new document. Doctors are granted access to do this through HLF.

- **Sharing with Insurers:** Insurers are very similar to providers in way of sharing, but insurers channel provides them a way to see history of records. Also, Insurers can ask for records approvals from providers as well.

- **Sharing with Researchers:** Researchers are less trusted stakeholders than providers and insurers. Patients do not want to share any personal information with researchers and they are not provided with any nucypher key set upon registration. Channel with patient and researchers holds a format where, patient generate a nucypher key set and create a share policy to share data upon that channel.

  Any interested researcher may use that key set to access data. This way it removes any additional effort that patient has to make for sharing EHR data.

3.5 MedFabric4Me System Deployment in Current HIE Setting

Currently HIE systems are cloud based and centralised. Different providers and health organizations request HIE to participate and trust centralised cloud security provided by HIE for communication.

MedFabric4Me is a decentralised system, which means there are multiple nodes at which providers, patients and other healthcare organizations can register. MedFabric4Me handles access control via smart contracts whereas certificate authorities controls participation. Key difference between HIE and MedFabric4Me architecture is enrollment criteria. MedFabric4Me can provide different criteria of enrollment for different organizations, like a provider may need to provide some medical information while patients need to provide social identity.

Other than that, HIE allows central connectivity to different Healthcare applications while MedFabric4Me allows different applications and data access based on nodes. Every node can host a client application and provide multiple options of enrollment for participating actors.

Modular architecture of MedFabric4Me also allows it do horizontally expand in features. MedFabric4Me could help researchers and insurer with data analytics and claims data. Other than that, MedFabric4Me could also register new nodes dynamically, thus providing scalability to easily on board new organizations.

MedFabric4Me needs to host ordering service and peer nodes. Flexible design of MedFabric4Me allows multiple physical machine ordering service and distributed node system. This allows a more evenly distributed cost architecture among different organizations. HIE can host ordering services and providers/insurers/researchers can host multiple nodes, managing on their own. Similarly, Providers can host patient nodes as well, allowing them to provide their own authorization methods for patients.

Chapter 4

EXPERIMENTS AND RESULTS

4.1 Description of Experiments

The main purpose of experiments is to assess MedFabric4Me with a state of art system. For this purpose we have implemented MedFabric4Me as minimal-viable product and tested its performance against healthcurrent, an HIE working in Arizona. Healthcurrent has more than 500 participating organizations and operates all over Arizona. Healthcurrent statistics data is available online on their website which we will be using as baseline performance measure for MedFabric4Me.

4.1.1 Comparison Matrix

Healthcurrent handles multiple types of healthcare transactions and alerts. Some key transactions are :

- **Health Level Seven (HL7):** HL7 refers to a set of international standards for transfer of clinical and administrative data between software applications used by various healthcare providers. These standards focus on the application layer, which is "layer 7" in the Open Systems Interconnection model. The HL7 standards are produced by Health Level Seven International, an international standards organization, and are adopted by other standards issuing bodies such as American National Standards Institute and International Organization for Standardization.

- **Continuity of Care Document (CCD):** A CCD is an XML-based markup standard intended to specify the encoding, structure and semantics of a patient

summary clinical document for exchange. It provides a means for one health care practitioner, system, or setting to aggregate pertinent data about a patient and forward it to another practitioner, system or setting to support the continuity of care.

- **Alerts:** Alerts are sent to designated clinicians or individuals based upon a patient panel. A patient panel is a practice or payer provided list of patients/members they wish to track.

For comparison, we would be comparing our HLF throughput with average number of transactions(HL7, CCD and Alerts) in Healthcurrent. Comparison between different number of peers would provide scalability of our system.

Other than that, success in HIE arose from the field of cloud computing. To compare performance of IPFS, comparison between upload and download speed of cloud services should be a good measure for performance.

Nucypher is a PRE application which we compared with AES256 encryption, that is most commonly used for cloud encryption.

4.2 Actors in MedFabric4Me

The application of MedFabric4Me spans over four functional units. These functional units are called stakeholders when referenced in the context of the application:

1. **Patients:** Data owner, controls sharing and Nucypher policy creation.
2. **Provider:** Generates EHRs, can update or delete upon consent.
3. **Insurer:** Insurance providers, need data for verification, can't update but request further information from provider.
4. **Researcher:** Can't update or request. Can only read EHR upon consent.

A participant or a physical organization may assume multiple roles in our application. For example, an insurance company can register for Insurer as well as Researcher.

4.3 Experimental Data

Healthcare data is highly sensitive and protected by HIPAA. For the privacy of patients, de-identified data is used, which in any way can't be associated with any entity. Provided data covers multiple categories of medical records: structured, unstructured and images as well.

For the purpose of testing, this data was emulated is different ways and some sampled data was created for the experiments.

4.3.1 Description of EHR Data Management

The Healthcare industry developed in silos of Providers and insurers, and never had any universal defined standard for EHRs. This forces us to create a system that can take a wide variety of data whether structured or unstructured data. EHRs consist of images, written notes, compressed files, videos and audios, other than standard structured data.

To manage all this, we deviated from traditional databases and chose IPFS. IPFS can store any kind of data and provides a unique hash value of saved files based on it's contents. It can create hash from text files, images and even videos if provided as byte string.

For our application, we used de-identified medical data that consist of clinical visits, diagnosis and conditions, procedures, medications, vitals and MRI images. Some of the data is structured like medications and vitals while some of it is unstructured like clinical visits, which may contains medical examiner's notes.

4.3.2 *Managing Records*

Medical records are divided in two types, structured and unstructured.

- **Structured Data:** Structured data has fixed columns and can easily be converted into JSON files. This way it is easier to create a merkle tree for each data entry and share partial data from the document. For example: Vitals has fixed columns, name, value and date. This way we can create a file with merkle tree and share only vital values that are required.

  This need-to-know basis sharing is quite useful for de-identifying documents.

- **Unstructured Data:** Unstructured data can't be converted directly into merkle tree. So, unstructured data is directly stored and we can't share partial data with merkle proof from these records, Although, IPFS can read these text/images file and makes sure to provide unique hash values.

- **Encrypted Files:** Nucypher encrypts files and then saves them on IPFS, which can not be read by IPFS to create hash. IPFS then directly creates a hash value from file of name which could coincide with other encrypted names as well. To solve this problem,the files are named by hashing some document variables, time based variables and patient public keys to make name as unique as possible.

4.4 Hyperledger Fabric System

HLF is a permissioned blockchain network and consists of multiple important components. The infrastructure consists of the following components:

1. **Peers:** Peers are a fundamental element of the network because they host ledgers and smart contracts. For redundancy, resilience and reliability, every

organisation hosts multiple peers. These peers maintain copies of the shared distributed ledgers that the peers are part of. We will vary number of peers for our performance testing for scalability check.

2. **Client Peers:** Client Peers host the REST Application Programming Interface, that allows user applications to interact with the smart contracts. A client application talks to a client peer over REST or GRPC interface and submits transactions and queries to the peer.

3. **Certification Authority (CA):** Every organization hosts a certificate authority(CA) which provides features such as:

   - Registration of identities
   - Issuing Enrollment Certificates(ECerts)
   - Certificate renewal and revocation

4. **CouchDB:** For a HLF blockchain network, the current state of the ledger represents the latest values for all keys included in the chain transaction logs. This is commonly referred to as World State. Chaincode invocations execute transactions against the current state data. To make chaincode interactions efficient, the latest values of all keys are stored in state database, which is essentially an indexed view of the blockchain's transaction log. The state database is recovered or generated upon peer startup automatically. LevelDB is default database embedded in the peer process and stores chaincode data as key-value pairs.

   CouchDB is an optional alternative database that provides additional query support for chaincode data modelling. In this implementation, CouchDB is hosted for every organization, which allows running rich queries on the data stored on the distributed ledger. This application needs searches based on

Patient's ID or Provider ID, which could speed up the performance for querying purposes.

5. **Ordering Service:** The ordering service is implemented using Apache Kafka and RAFT. For Kafka based ordering service, Orderer consist of a Zookeeper cluster consisting of three Zookeeper nodes. On top of the Zookeeper cluster is a cluster of Kafka brokers. These Kafka brokers replicate transactions that are set in order, according to timestamps by three orderer nodes.

   Whereas with RAFT based fault tolerant consensus, it is easier to spin up a orderer node. RAFT does not third party dependant zookeeper and kafka nodes.

   We will compare these two consensus algorithms for performance and conclude whether it is good to move to RAFT from Kafka.

6. **Channels:** Four channels are used to enable transactions on our applications. Channels are created to share data effectively and with relevant organizations only.

4.4.1 Hyperledger Caliper

Hyperledger Caliper[4] is an open-source blockchain performance framework, which allows users to test different blockchain solutions with predefined use cases and configurations. It can be used for comparative performance studies across different blockchain technologies, performance testing for smart contracts and discovering resource constraints for test loads. Figure 4.1 shows the architecture for Hyperledger Caliper.

The configuration includes information about the benchmark to be run, the blockchain architecture to be tested and the code for the smart contracts. These are fed to the interface, which creates two types of clients. The Admin client configures the sys-

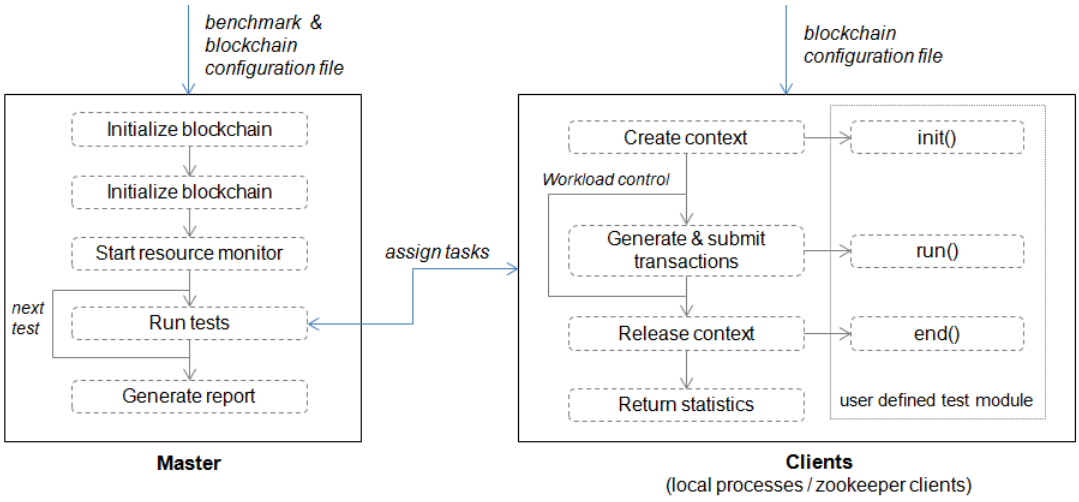


Figure 4.1: Hyperledger Caliper Architecture[1]

tem under test. This client is used to create channels, allow peers to join channels and deploy chaincode. Multiple worker clients are also created which drive the test load. These clients are driven using a rate controller, which impacts the rate at which transactions are submitted. The responses from these transactions are published to a performance analyzer. Resource monitor keeps track of the memory usage, disk I/Os and CPU usage. All of these statistics are then compiled into a report.

4.4.2 Performance Testing

Testbed

HLF scalability is very important for application. So, we tested HLF with varying number of peers against provided two Ordering services : Kafka and RAFT. Table 4.1 contains all the parameter values that remain constant across all tests. The first phase of experiments involves performance testing the blockchain network using the following configurable parameters:

- **MaxMessageCount:** Maximum number of Transactions allowed in a block.

Table 4.1: Fixed Parameters During Performance Testing

Parameter Name	Value
Max Block Size	98 MB
No. of Orderers	3
System Memory (Total)	236 GB
OS	Ubuntu 16.04.3 LTS

- **Preferred Max Bytes:** The size of the blocks that are replicated across the network.
- **Send Rate:** Rate at which transactions are sent as input to the blockchain network during the experiment. Every experiment has multiple rounds which with increasing send rates and performance metrics are recorded and reported for each round.

For performance purposes, Arizona based HIE, healthcurrent has over 674 hospitals, 91k patients and process over 23.4 million requests per month(Average). To simulate a network of that size around Arizona, we will be testing our application over upto 50 peers. But directly scaling upto that size doesn't allow us to verify the scaling along the way.

Table 4.2: Number of Peers During Testing

Patient	Provider	Insurer	Researcher	Total
7	7	3	3	20
15	15	5	5	40
20	20	5	5	50

The performance metrics for the first set of experiments can be described as follows:

- **Max Latency:** Worst Case Transaction Response Time (measured in seconds).
- **Min Latency:** Best Case Transaction Response Time (measured in seconds).
- **Avg Latency:** Mean Transaction Response Time (measured in seconds).
- **Throughput:** Transactions Per Second processed by the system.
- **Successful Transactions:** Percentage of Transactions that were successful.
- **Failed Transaction:** Percentage of Transactions that were unsuccessful.

Each experiment contains details, results and discussion for testing the underlying blockchain infrastructure using different consensus services. Hyperledger caliper carries out this test by sending fixed number of transactions at varying transaction rates.

Orderer: Kafka

In Kafka setup of HLF, Ordering service has 3 orderers, 3 Zookeeper nodes and 4 kafka nodes. Kafka needs 3n+1 nodes to manage n faults.

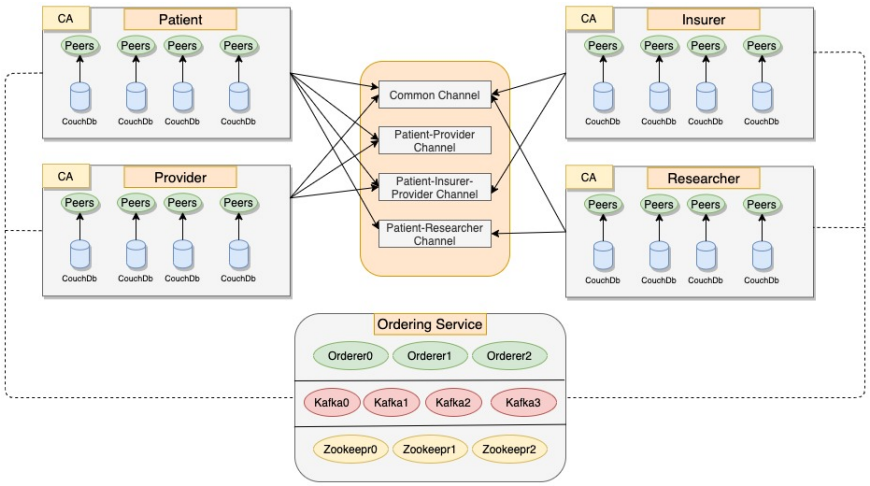


Figure 4.2: Kafka Hyperlerdger Structure

Peer Size 20

Adding Asset to Blockchain Report generated by caliper consists how many transactions got successful and throughput for transactions.

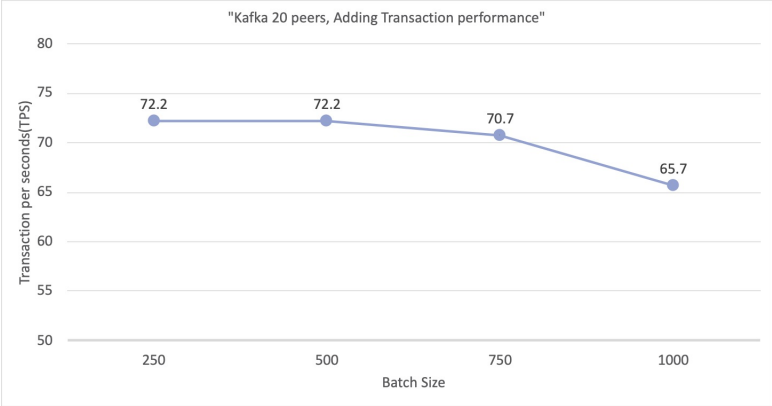


Figure 4.3: Hyperledger Fabric Kafka Network with 20 Peers; Adding Asset Performance

Table 4.3: Performance Results for Adding Assets; 20 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.1	0.54	0.12	0.33	24.8
	100	0	50.2	0.56	0.13	0.27	48.9
	100	0	100.4	2.21	0.47	1.58	69.6
	100	0	163.1	2.94	1.27	2.42	70.1
	100	0	345.6	3.37	1.72	2.40	72.2
500	100	0	25.1	0.73	0.12	0.33	24.9
	100	0	50.1	0.49	0.12	0.26	49.5
	100	0	100.2	5.09	0.34	3.45	66.6
	100	0	200.5	5.57	1.29	4.94	70.4
	100	0	386.4	6.65	4.34	5.29	72.2
750	100	0	25.0	0.64	0.12	0.34	24.8
	100	0	50	0.53	0.13	0.27	49.6
	100	0	99.7	7.74	0.32	5.14	68.4
	100	0	200.4	8.28	0.56	7.10	70.7
	100	0	305.5	9.85	2.19	8.03	69.0
1000	100	0	25.0	0.68	0.12	0.35	24.9
	100	0	50.1	0.60	0.14	0.28	49.6
	100	0	100.2	10.98	0.43	7.77	64.3
	100	0	192.6	11.96	0.69	10.21	65.1
	100	0	395.1	14.00	7.89	11.27	65.7

Querying Ledger To query ledger we used the same rounds of tps and same configuration. Rather than invoking chaincode which requires endorsement from other peers, we will just query the ledger, which doesn't need endorsement.

Results With above graphs and performance matrices, we can easily observe that :

- Overall latency for either adding assets or querying the network was low. This is expected due to low number of peers.
- Throughput among Adding assets is consistent among all batch sizes. While,

Table 4.4: Performance Results for Querying Assets; 20 Peers Network

Batch Size	**Success (%)**	**Fail (%)**	**Send Rate**	**Max Latency**	**Min Latency**	**Avg Latency**	**Throughput (TPS)**
250	100	0	25.1	0.03	0.01	0.01	25.1
	100	0	50.2	0.03	0.01	0.01	50.0
	100	0	100.4	0.13	0.01	0.04	99.8
	100	0	201.3	0.36	0.05	0.18	192.6
	100	0	355.6	1.27	0.73	0.99	173.3
500	100	0	25.1	0.08	0.01	0.01	25.0
	100	0	50.1	0.10	0.01	0.01	50.0
	100	0	100.2	0.30	0.01	0.04	99.9
	100	0	195.6	1.37	0.04	0.69	179.1
	100	0	359.7	1.54	0.28	1.14	227.7
750	100	0	25.0	0.11	0.01	0.01	25.0
	100	0	50.1	0.12	0.01	0.02	50.0
	100	0	100.1	0.26	0.01	0.04	99.9
	100	0	200.4	0.98	0.05	0.42	193.1
	100	0	370.4	2.55	0.35	2.04	222.3
1000	100	0	25.0	0.18	0.01	0.02	25.0
	100	0	50.1	0.21	0.01	0.02	50.0
	100	0	100.1	0.44	0.01	0.05	99.9
	100	0	200.2	1.89	0.03	0.89	195.9
	100	0	394.9	3.27	0.68	2.52	234.5

querying steadily increases with a slight drop for 750 batch size. This concludes that general adding or querying assets do not get effected much due to higher batch sizes in lower number of peers.

- There are no transaction failures at any batch size. This shows that Kafka seems to adapt very well for low number of peers.

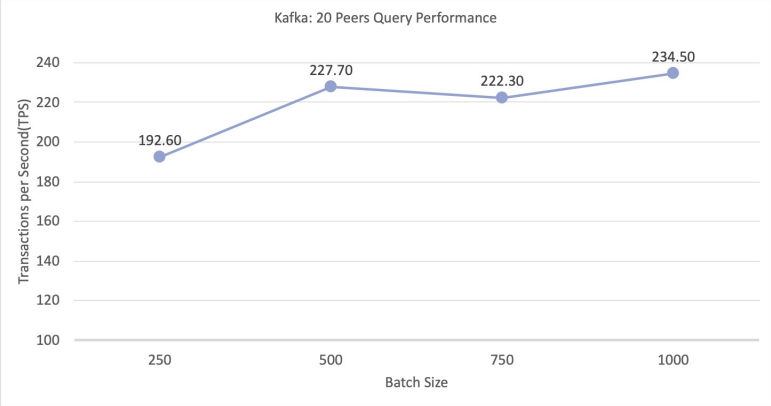


Figure 4.4: Hyperledger Fabric Kafka Network with 20 Peers; Query Performance

Peer Size 40

Adding Asset to Blockchain Table 4.5 consists of results performed by Hyperledger caliper on peer size 40 network with kafka ordering service for adding assets.

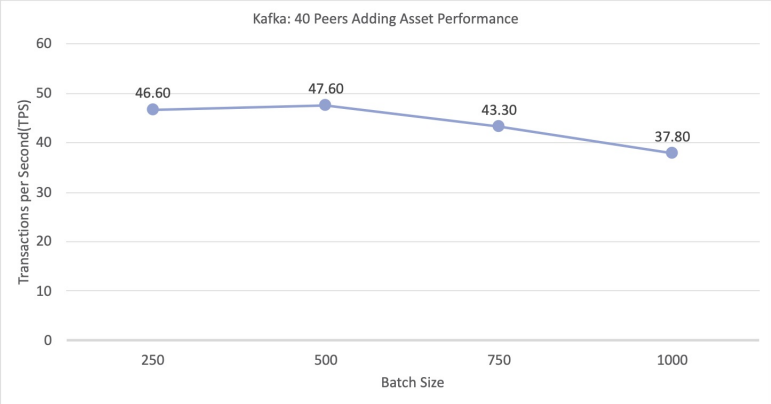


Figure 4.5: Kafka Network with 40 Peers; Adding Asset Performance

Querying Ledger Table 4.6 consists of results performed by Hyperledger caliper on peer size 40 network with kafka ordering service for querying assets.

Table 4.5: Performance Results for Adding Assets; 40 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.0	0.51	0.10	0.32	24.9
	100	0	50.1	0.79	0.12	0.41	29.6
	100	0	100.1	7.19	0.21	5.49	44.5
	100	0	193.4	12.31	0.74	9.10	46.6
	100	0	395	17.78	3.58	10.67	42.1
500	100	0	24.9	0.81	0.13	0.51	24.4
	100	0	49.9	0.93	0.14	0.45	45.9
	100	0	100	8.82	0.65	5.39	47.6
	100	0	198.8	13.49	1.15	11.24	44.2
	100	0	376.1	19.87	4.80	16.21	45.6
750	100	0	25.0	0.78	0.16	0.43	24.9
	100	0	50	0.84	0.23	0.67	37.9
	100	0	100.1	9.97	0.30	6.46	40.6
	100	0	197	16.94	1.27	13.85	43.3
	100	0	364	17.20	2.75	15.38	38.5
1000	100	0	25.0	0.56	0.10	0.28	24.5
	100	0	50.1	0.82	0.12	0.54	37.8
	100	0	100.1	17.48	0.72	12.80	24.3
	45.1	54.9	200.2	15.63	2.84	12.15	18.8
	62.2	37.8	385	18.28	6.50	15.24	26.9

Results With above graphs and performance matrices, we can easily observe that :

- Overall latency for either adding assets or querying the network is higher than 20 peer network. This concludes that due to higher number of peers to keep track of latency increases.
- Throughput among Adding assets consistently drops with increase in batch size. While, querying also steadily decreases with a slight uplift for 750 batch size. This suggests that with increase in peer numbers, batch sizes effects inversely to transactions per seconds.

Table 4.6: Performance Results for Querying Assets; 40 Peers Network

Batch Size	**Success (%)**	**Fail (%)**	**Send Rate**	**Max Latency**	**Min Latency**	**Avg Latency**	**Throughput (TPS)**
250	100	0	25.1	0.03	0.01	0.01	25.1
	100	0	50.1	0.08	0.01	0.02	50.1
	100	0	100.2	0.26	0.02	0.28	99.4
	100	0	201.3	0.68	0.08	0.45	172.8
	100	0	386.2	1.35	0.65	0.86	164.7
500	100	0	25.1	0.08	0.01	0.01	25.0
	100	0	50.1	0.06	0.01	0.02	50.1
	100	0	100.1	0.42	0.01	0.14	100.0
	100	0	199.9	1.67	0.04	0.74	159.4
	100	0	345.2	2.62	0.36	1.73	154.5
750	100	0	25.0	0.10	0.01	0.02	25.0
	100	0	50.1	0.67	0.01	0.02	50.0
	100	0	100.1	0.30	0.01	0.04	99.9
	100	0	200.2	1.80	0.02	1.92	158.5
	100	0	374.6	2.62	0.42	1.94	167.3
1000	100	0	25.1	0.12	0.01	0.02	25.0
	100	0	50.0	0.18	0.01	0.02	50.0
	100	0	100.0	1.16	0.01	0.18	99.9
	100	0	200.2	2.98	0.02	2.11	146.9
	100	0	397.0	2.93	0.86	2.26	153.2

- There are transaction failures at higher batch sizes. This shows that due to higher number of transactions along with high transactions per second, endorsement fails sometimes due to inability of ordering service to perform at such speed.

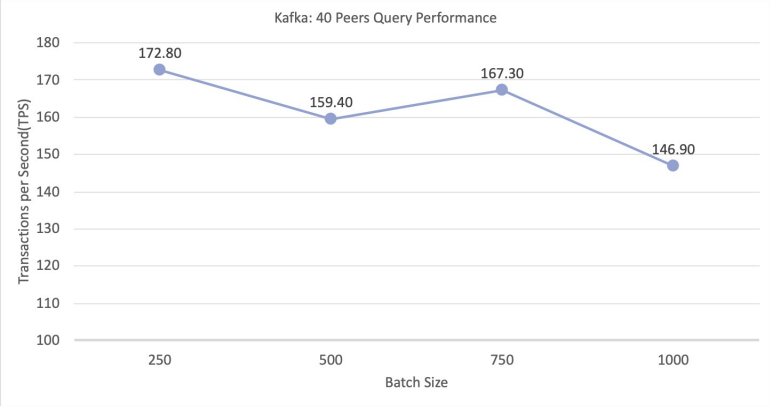


Figure 4.6: Hyperledger Fabric Kafka Network with 40 Peers; Query Performance

Peer Size 50

Adding Asset to Blockchain Table 4.7 consists of results performed by Hyperledger caliper on peer size 50 network with kafka ordering service for adding assets.

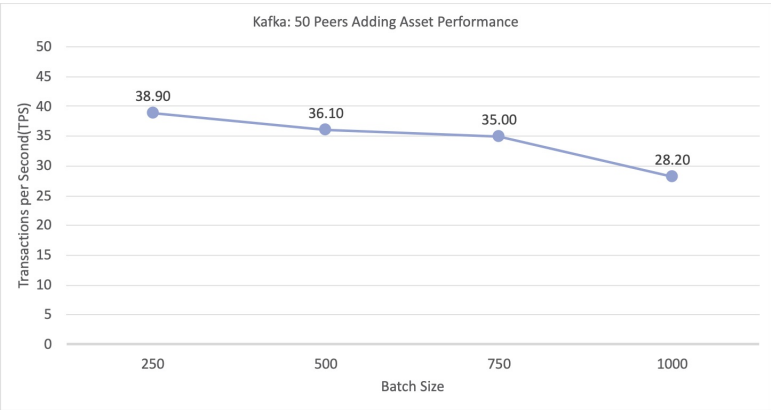


Figure 4.7: Hyperledger Fabric Kafka Network with 50 Peers; Adding Asset Performance

Querying Ledger Table 4.8 consists of results performed by Hyperledger caliper on peer size 40 network with kafka ordering service for querying assets.

Table 4.7: Performance Results for Adding Assets; 50 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.1	0.68	0.14	0.37	24.3
	100	0	50.2	0.95	0.15	0.35	37.9
	100	0	100.4	9.91	0.28	7.94	38.9
	100	0	197.2	14.10	0.83	11.65	37.4
	100	0	216.1	19.00	4.62	11.24	34.6
500	100	0	25.0	0.89	0.17	0.54	24.5
	100	0	48.6	1.23	0.24	0.67	36.1
	100	0	100.2	8.57	0.45	7.34	35.7
	100	0	200.5	15.10	1.32	13.10	35.1
	100	0	242.1	21.24	6.80	18.43	34.5
750	100	0	25.0	0.68	0.14	0.37	25.1
	100	0	50.1	1.20	0.43	0.76	35.0
	100	0	100.1	10.46	0.34	7.94	34.3
	100	0	200.3	18.40	1.14	11.65	34.6
	100	0	357.7	21.37	3.8	16.68	33.7
1000	100	0	25.0	0.62	0.12	0.34	23.5
	79.6	20.4	50	0.74	0.15	0.43	28.2
	22.6	77.4	100.1	8.05	1.28	6.41	15.7
	30.6	69.4	165.5	18.58	3.23	13.56	18.5
	28.6	71.4	306.9	19.56	8.21	15.28	11.3

Results With above graphs and performance matrices, we can easily observe that :

- Overall latency for either adding assets or querying the network is higher than previous setups. This supports our results we concluded in 40 peer setup.
- Throughput among Adding assets consistently drops with increase in batch size and, querying also steadily decreases. This supports our results we concluded in 40 peer setup.
- There are very commonly occurring transaction failures at higher batch sizes. This shows that in large networks, higher number of transactions along with

Table 4.8: Performance Results for Querying Assets; 50 Peers Network

Batch Size	**Success (%)**	**Fail (%)**	**Send Rate**	**Max Latency**	**Min Latency**	**Avg Latency**	**Throughput (TPS)**
250	100	0	25.1	0.32	0.01	0.03	25.1
	100	0	50.2	0.59	0.01	0.07	50.1
	100	0	100.4	0.66	0.02	0.37	99.6
	100	0	201.3	1.06	0.16	0.63	157.5
	100	0	355.6	1.61	0.85	1.21	145.5
500	100	0	25.1	0.09	0.01	0.01	25.0
	100	0	50.1	0.10	0.01	0.02	50.0
	100	0	100.2	0.69	0.01	0.12	99.8
	100	0	200.3	2.38	0.10	1.30	142.9
	100	0	324.5	3.00	0.26	2.34	128.1
750	100	0	25.0	0.09	0.01	0.02	25.0
	100	0	50.1	0.43	0.01	0.03	50.0
	100	0	100.1	0.46	0.01	0.08	96.5
	100	0	200.2	3.00	0.04	2.25	129.6
	100	0	389.2	3.10	0.26	2.69	104.3
1000	100	0	25.0	0.12	0.01	0.02	25.0
	100	0	50.1	0.18	0.01	0.03	50.0
	100	0	100.1	1.17	0.01	0.18	99.8
	97.4	2.6	200.4	3.00	0.04	2.19	160.9
	90.02	9.98	397.5	3.11	0.18	0.26	69.6

high transactions per second, endorsement fails more frequently due to inability of ordering service to perform at such speed.

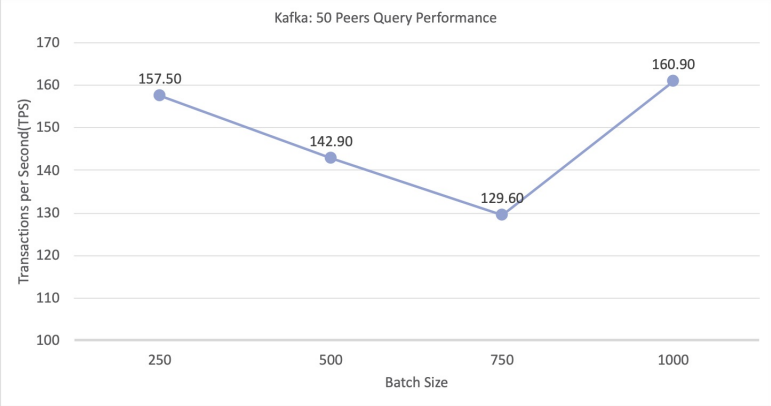


Figure 4.8: Hyperledger Fabric Kafka Network with 50 Peers; Query Asset Performance

Orderer: RAFT

In RAFT setup of HLF, Ordering service has 1 leader orderer and 3 follower orderers. RAFT only needs 2n+1 nodes to manage n faults.

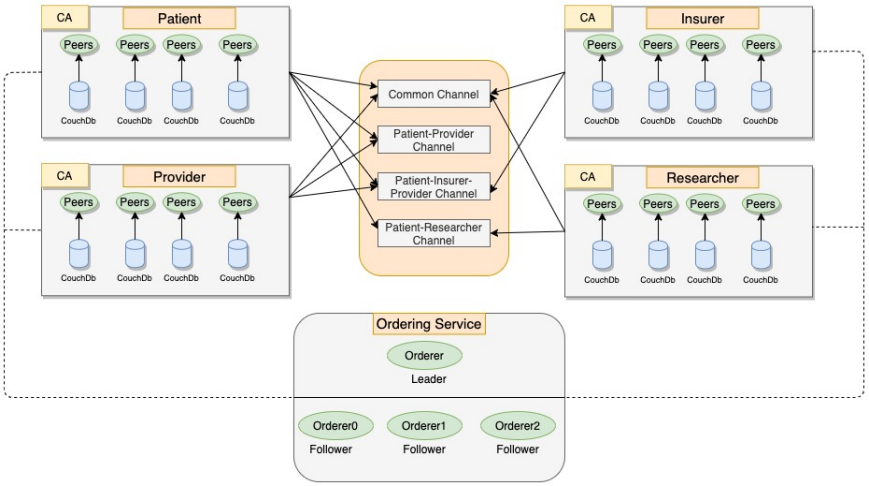


Figure 4.9: RAFT Hyperlerdger Structure

Peer Size 20

Adding Asset to Blockchain Table 4.9 consists of results performed by Hyperledger caliper on peer size 20 network with RAFT ordering service for adding assets.

Table 4.9: Performance Results for Adding Assets; 20 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.1	2.87	0.11	0.99	24.6
	100	0	50.2	5.16	0.12	3.43	27.5
	100	0	100.6	2.96	0.14	2.10	61.8
	100	0	189.2	4.34	0.21	2.85	45.4
	100	0	291.0	4.32	2.77	3.64	45.4
500	100	0	25.1	5.15	0.12	2.30	24.2
	100	0	50.1	4.99	0.12	3.41	39.1
	100	0	87.4	6.05	2.93	4.71	43.1
	100	0	187.8	6.84	0.26	5.02	45.8
	100	0	205.8	6.36	0.52	4.70	53.2
750	100	0	25.0	5.23	0.12	1.82	24.8
	100	0	50.1	5.09	0.14	2.05	39.5
	100	0	88.8	7.02	0.22	4.62	64.0
	100	0	191.2	6.59	0.35	4.41	60.5
	100	0	304.3	8.20	0.23	5.81	61.8
1000	100	0	25.0	5.21	0.13	1.99	24.9
	100	0	50.1	5.23	0.18	3.35	43.0
	100	0	86.4	7.96	0.60	5.42	59.8
	100	0	192.0	10.88	0.37	5.63	59.3
	12.6	87.4	286.8	11.55	0.29	7.25	3.8

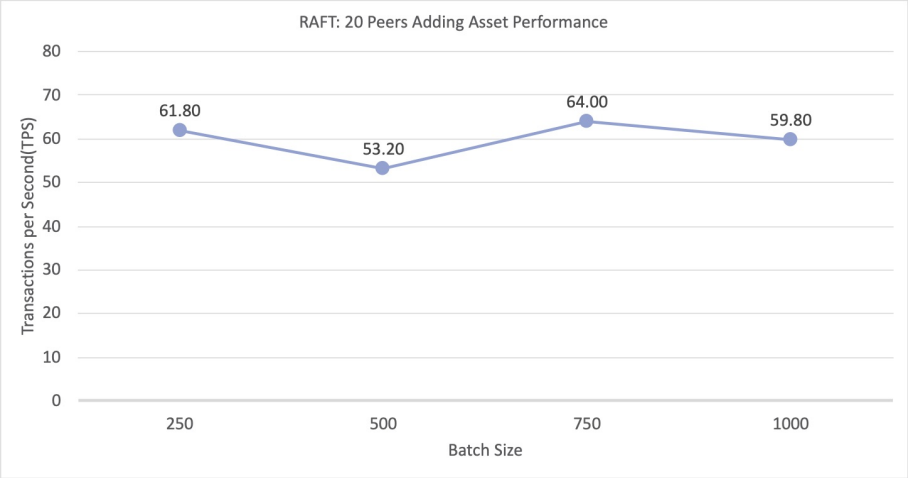
Querying Ledger Table 4.10 consists of results performed by Hyperledger caliperon peer size 20 network with RAFT ordering service for querying assets.

Table 4.10: Performance Results for Querying Assets; 20 Peers Network

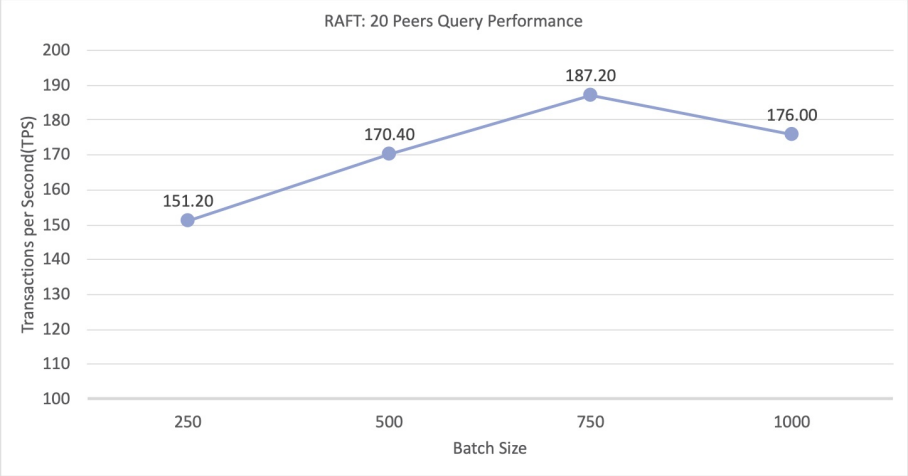
Batch Size	**Success (%)**	**Fail (%)**	**Send Rate**	**Max Latency**	**Min Latency**	**Avg Latency**	**Throughput (TPS)**
250	100	0	25.1	0.57	0.01	0.05	25.1
	100	0	50.2	0.13	0.01	0.03	50.0
	100	0	100.4	0.44	0.01	0.09	97.0
	100	0	201.3	1.33	0.02	0.71	151.2
	100	0	363.4	1.58	0.34	1.11	142.8
500	100	0	25.1	0.08	0.01	0.04	25.0
	100	0	50.1	0.42	0.01	0.04	50.0
	100	0	100.2	0.66	0.01	0.08	98.9
	100	0	200.7	1.79	0.02	0.85	167.1
	100	0	354.6	2.66	0.04	2.05	170.4
750	100	0	25.0	0.76	0.01	0.02	25.0
	100	0	50.1	0.50	0.01	0.03	49.9
	100	0	100.1	0.76	0.01	0.09	100.0
	100	0	196.9	3.78	0.02	2.22	151.5
	100	0	370.9	3.65	1.96	2.58	187.2
1000	100	0	25.0	0.11	0.01	0.02	25.0
	100	0	50.1	0.12	0.01	0.03	50.0
	100	0	100.1	0.20	0.01	0.07	100.0
	100	0	200.4	6.97	2.51	4.82	131.9
	100	0	385.7	4.71	3.06	3.83	176.0

Results With above graphs and performance matrices, we can easily observe that :

- Overall latency for either adding assets or querying is higher than ususal. This is due to the fact that RAFT ordering service needs TLS certificates of peers before committing any query.
- Throughput among Adding assets is consistent while for query it increases gradually.
- There are very minimal transaction failures.



(a) Adding Assets Performance



(b) Query Performance

Figure 4.10: Hyperledger Fabric Network with 20 Peers and RAFT Ordering Service

Peer Size 40

Adding Asset to Blockchain Table 4.11 consists of results performed by Hyperledger caliper on peer size 40 network with RAFT ordering service for adding assets.

Table 4.11: Performance Results for Adding Assets; 40 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.0	2.12	0.04	0.78	24.9
	100	0	50.1	2.46	0.12	2.10	27.4
	100	0	100.6	8.60	0.16	7.68	38.8
	100	0	191.2	16.74	1.26	12.47	42.3
	100	0	306.2	21.51	5.32	12.88	32.1
500	100	0	25.0	4.38	0.08	3.97	24.6
	100	0	49.9	5.82	0.18	4.78	36.7
	100	0	100.1	11.23	0.14	8.37	34.3
	100	0	176.2	14.82	1.16	11.43	36.1
	100	0	314.4	22.36	6.77	17.48	27.8
750	100	0	25.1	3.23	0.04	2.31	24.9
	100	0	50.0	6.12	2.38	4.68	34.6
	100	0	100.0	14.33	2.12	11.37	28.2
	100	0	191.4	16.46	4.11	12.14	30.1
	100	0	300.2	28.96	5.76	22.71	23.4
1000	100	0	25.0	2.32	0.48	1.73	24.9
	100	0	50.1	14.66	0.42	11.82	27.8
	100	0	100.1	24.32	2.13	21.47	24.3
	94.6	0.4	200.2	21.69	1.89	17.28	18.9
	67.4	32.6	229	29.54	6.10	26.44	21.3

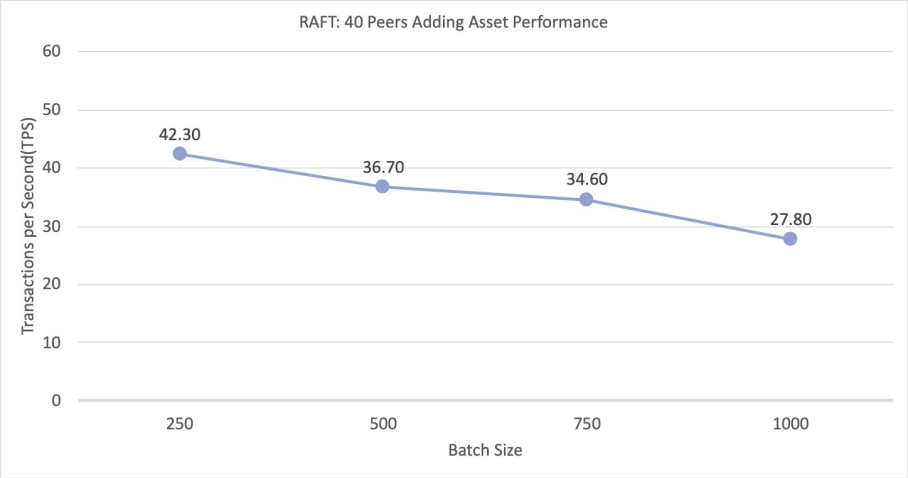
Querying Ledger Table 4.12 consists of results performed by Hyperledger caliper on peer size 40 network with RAFT ordering service for querying assets.

Results With above graphs and performance matrices, we can observe that :

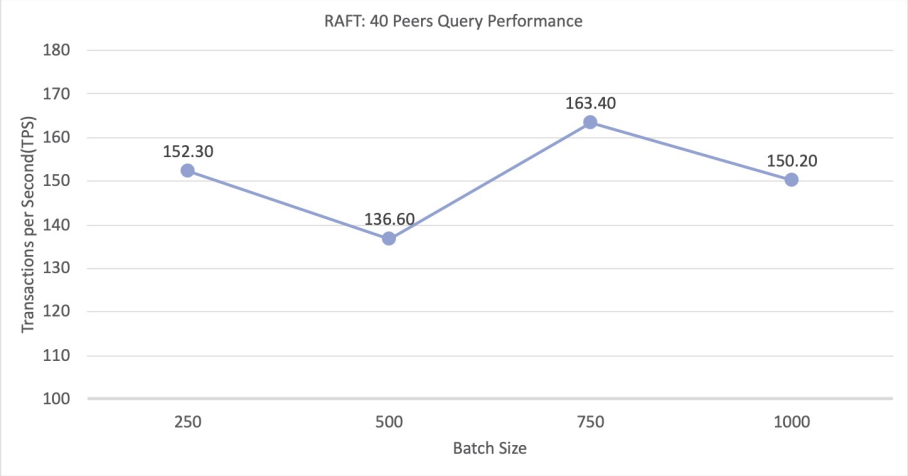
Table 4.12: Performance Results for Querying Assets; 40 Peers Network

Batch Size	**Success (%)**	**Fail (%)**	**Send Rate**	**Max Latency**	**Min Latency**	**Avg Latency**	**Throughput (TPS)**
250	100	0	25.0	0.62	0.01	0.07	25.1
	100	0	50.1	0.56	0.01	0.11	50.0
	100	0	100.4	0.76	0.01	0.14	97.2
	100	0	201.3	1.24	0.04	0.57	152.3
	100	0	355.6	1.84	0.24	1.36	142.6
500	100	0	25.1	0.05	0.01	0.04	25.0
	100	0	50.1	0.36	0.01	0.03	50.0
	100	0	100.2	0.85	0.01	0.09	98.7
	100	0	195.6	1.54	0.08	1.10	136.6
	100	0	359.7	3.24	0.13	2.19	125.4
750	100	0	25.0	1.21	0.01	0.04	25.0
	100	0	50.1	0.64	0.01	0.07	50.0
	100	0	100.1	0.56	0.01	0.09	100.0
	100	0	200.4	3.98	0.05	2.24	152.9
	100	0	370.4	3.78	1.53	2.40	163.4
1000	100	0	25.0	0.14	0.01	0.02	25.0
	100	0	50.1	0.11	0.01	0.04	50.0
	100	0	100.1	0.30	0.01	0.08	99.9
	100	0	200.2	3.73	0.64	2.46	148.2
	100	0	394.9	5.72	3.86	4.25	150.2

- Overall latency for either adding assets or querying is higher than usual. This supports our suggestion that latency increases with increase in number of peers.
- Throughput among Adding assets consistently drops while for querying, it remains consistent with some variations.
- There are very minimal transaction failures.



(a) Adding Assets Performance



(b) Query Performance

Figure 4.11: Hyperledger Fabric Network with 40 Peers and RAFT Ordering Service

Peer Size 50

Adding Asset to Blockchain Table 4.13 consists of results performed by Hyperledger caliper on peer size 50 network with RAFT ordering service for adding assets.

Table 4.13: Performance Results for Adding Assets; 50 Peers Network

Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.0	1.72	0.21	1.12	24.4
	100	0	50.0	4.32	0.18	3.93	31.6
	100	0	96.4	11.23	1.42	9.38	33.1
	100	0	196.3	16.23	1.76	14.37	29.8
	100	0	296.1	29.98	5.92	23.19	23.3
500	100	0	24.9	5.13	0.11	4.14	24.9
	100	0	50.1	6.43	0.14	5.37	32.1
	100	0	100.2	24.31	0.76	18.92	27.1
	100	0	186.3	21.67	2.43	16.73	21.4
	100	0	284.7	27.96	9.92	24.19	26.2
750	100	0	25.1	1.90	0.03	0.36	24.7
	100	0	50.1	4.38	1.67	3.24	24.9
	100	0	99.9	19.48	1.92	12.73	23.1
	89.2	10.8	198.8	24.68	2.48	19.37	28.8
	73.8	26.2	272.6	28.34	7.51	22.97	17.8
1000	100	0	25.1	1.25	0.33	0.59	24.8
	100	0	49.9	28.42	0.67	21.94	22.0
	82.4	17.6	97.3	29.98	1.92	25.14	21.7
	77.7	22.3	196.6	29.69	1.75	18.37	22.3
	45.6	54.4	248.0	29.98	5.92	23.19	13.3

Querying Ledger Table 4.13 consists of results performed by Hyperledger caliper on peer size 50 network with RAFT ordering service for querying assets.

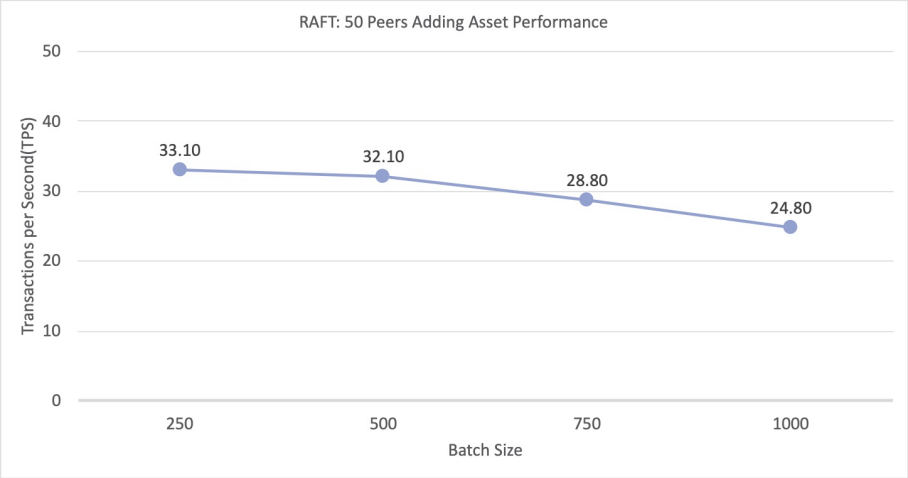
,

Table 4.14: Performance Results for Querying Assets; 50 Peers Network

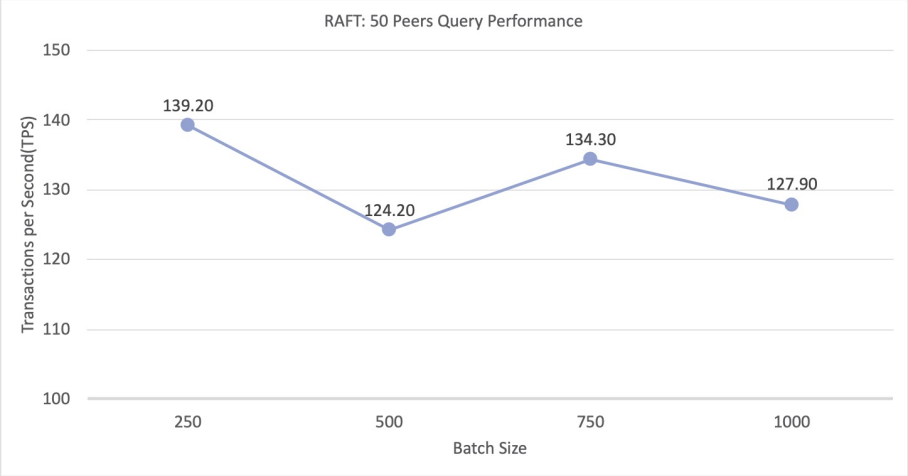
Batch Size	Success (%)	Fail (%)	Send Rate	Max Latency	Min Latency	Avg Latency	Throughput (TPS)
250	100	0	25.1	0.23	0.01	0.02	25.0
	100	0	50.0	0.41	0.02	0.04	50.0
	100	0	100.1	0.92	0.01	0.18	98.8
	100	0	201.0	1.32	0.06	0.63	107.1
	100	0	365.8	1.17	0.36	0.78	139.2
500	100	0	25.0	0.06	0.01	0.01	25.0
	100	0	50.2	0.14	0.01	0.02	50.1
	100	0	99.9	1.10	0.05	0.25	99.3
	100	0	200.0	1.76	0.07	1.19	124.2
	100	0	354.2	3.47	0.17	2.87	108.6
750	100	0	25.1	1.13	0.01	0.03	25.0
	100	0	50.2	0.75	0.01	0.08	50.0
	100	0	100.0	1.16	0.01	0.18	99.5
	100	0	200.1	2.67	0.04	2.12	102.8
	100	0	315.1	4.87	1.62	2.96	134.4
1000	100	0	25.0	0.12	0.01	0.02	25.0
	100	0	50.1	0.47	0.01	0.11	50.0
	100	0	100.1	3.71	0.07	0.13	99.1
	100	0	200.2	2.98	0.54	1.73	104.7
	100	0	357.0	4.15	2.81	3.69	127.9

Results With above graphs and performance matrices, we can observe that :

- Overall latency for either adding assets or querying is higher than previous setups. This supports our suggestion that latency increases with increase in number of peers.
- Throughput among Adding assets consistently drops while for querying, it remains consistent with some variations.
- There are very high transaction failures among large batch sizes.



(a) Adding Assets performance



(b) Query Performance

Figure 4.12: Hyperledger Fabric Network with 40 Peers and RAFT Ordering Service

4.5 IPFS

IPFS is a peer-to-peer hypermedia protocol. IPFS can have a private network or can be joined on their main decentralised network. Starting a node at IPFS public network can help to connect to multiple nodes without any memory or system constraint.

4.5.1 Benchmarking IPFS : js-ipfs Profiling

js-ipfs deamon to connect IPFS gateway is benchmarked using a profiling tool. Benchmarking tool is used to test upload and download speed of IPFS under different circumstances and different file sizes.

4.5.2 Performance Testing

Testbed IPFS can have a bottleneck performance due to peer-to-peer network, which usually provides low downloading speed. To test IPFS downloading and uploading speed, following tests and procedures are measured on files varying from 1 MB to 1GB:

- **Binary file Upload:** Uploading a document to IPFS which is not encrypted and IPFS system can read. IPFS needs to perform hashing to provide a access hash for this purpose.
- **Encrypted file upload:** As IPFS can't access this file, it should upload file without hashing.
- **Downloading:** IPFS download speed for files. Encryption shouldn't matter in this case.

For the testing purpose we spin up a node to connect public ipfs network and then

tested downloading and uploading speed. We used varied file sizes, from 1Mb to upto 1Gb, differing from JSON, xlsx and encrypted files.

IPFS performance is tested on public network and during the time of test there were 213 active nodes.

Performance Testing

Uploading Speed: Table 4.15 consists of results obtained by performing binary and encrypted file upload from server.

Table 4.15: IPFS Uploading Speed

File Size (MB)	Binary file (seconds)	Encrypted files(seconds)
1	0.46	0.41
100	2.66	0.41
200	4.95	0.40
300	7.39	0.40
400	9.81	0.42
500	13.75	0.42
600	19.83	0.42
700	25.44	0.42
800	37.59	0.43
900	46.74	0.43
1000	65.70	0.44

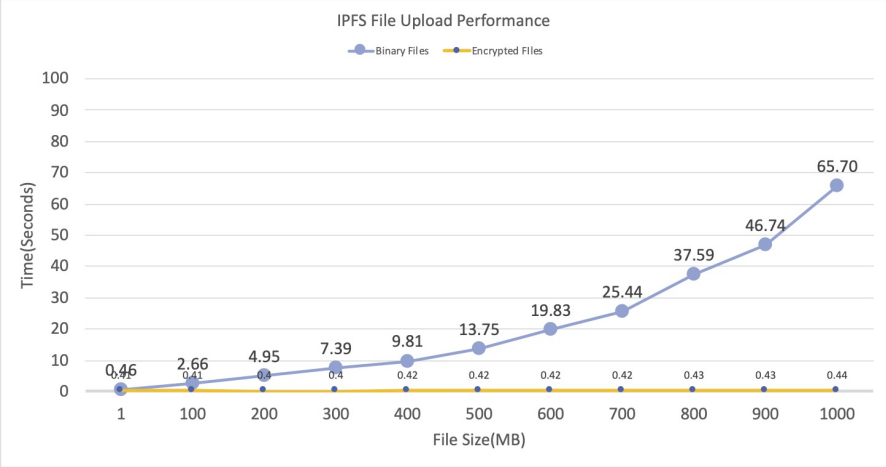


Figure 4.13: IPFS Uploading Speed

Downloading Speed: Table 4.16 consists of results obtained by performing binary file download from server.

Table 4.16: IPFS Downloading Speed

File Size (MB)	file download (seconds)
1	0.4
100	15.6
200	28.88
300	39.03
400	49.00
500	65.27
600	74.62
700	82.74
800	104.02
900	113.20
1000	134.72

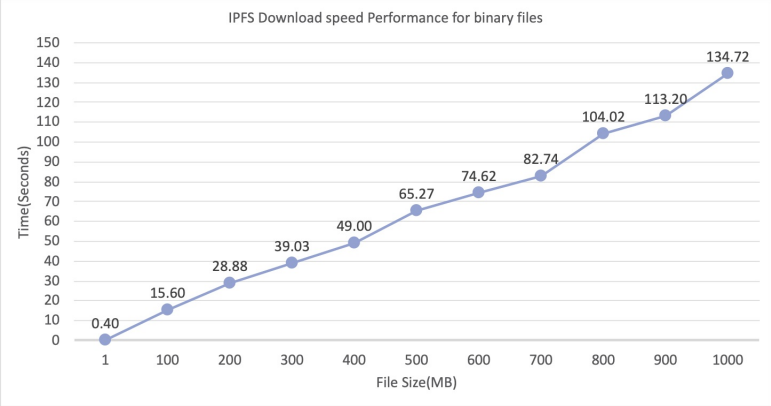


Figure 4.14: IPFS Downloading Speed

Results the above experiments shows that:

- As IPFS processes whole file to generate hash values, processing time is directly proportional to the size of record. While, IPFS can not process encypted files, it derives hash from it's name and processes it in constant time irrespective of it's size.
- Due to a p2p network, node discovery and downloading file takes more time than other centralised databases.

4.6 Nucypher

Nucypher system is a decentralised PRE application. It depends on four main actors:

1. **Alice:** Retains full control over the data encrypted for her and determines whom to share the data with.
2. **Bob:** The data recipient that Alice intends to share data with.

3. **Enrico:** A data source that encrypts data on behalf of Alice and produces a MessageKit.

4. **Ursula:** The nodes on the NuCypher Network that stand ready to re-encrypt data; they enforce the access policy created by Alice.

We will design a private Nucypher network with multiple Ursula nodes and check different functions performed by Nucypher network to observe bottleneck performance.

4.6.1 Timeit

Timeit is a python based benchmarking library that helps us to benchmark Nucypher's different functions under different conditions.

Testbed Nucypher is a decentralised PRE system, which relies on it's network of nodes. To spin up a network for scalability testing, we created a network of 50 nodes. Nucypher consist of multiple actors and particularly perform two costly actions :

1. **Encryption:** Nucypher does PRE with the help of enrico (Data encryptor) and key management through ursula. Alicia,(Data owner) encrypt data and then share policy with Bob(Data recipient).

2. **Decryption:** Data has been encrypted in a way that only Bob can access it. Bob need a combination of public- private key and fragmented key's from ursula to open the specific record.

Performance Testing

Results Nucypher even being decentralised PRE service, shows good performance for encryption and decryption.

Table 4.17: Nucypher Decryption Time

File Size (MB)	Decryption Time (seconds)	Encryption Time (seconds)
1	0.01	0.11
100	0.88	0.78
200	1.82	1.50
300	2.67	2.18
400	3.55	2.85
500	4.43	3.55
600	5.32	4.29
700	6.22	4.95
800	7.12	5.65
900	7.98	6.31
1000	8.68	7.11

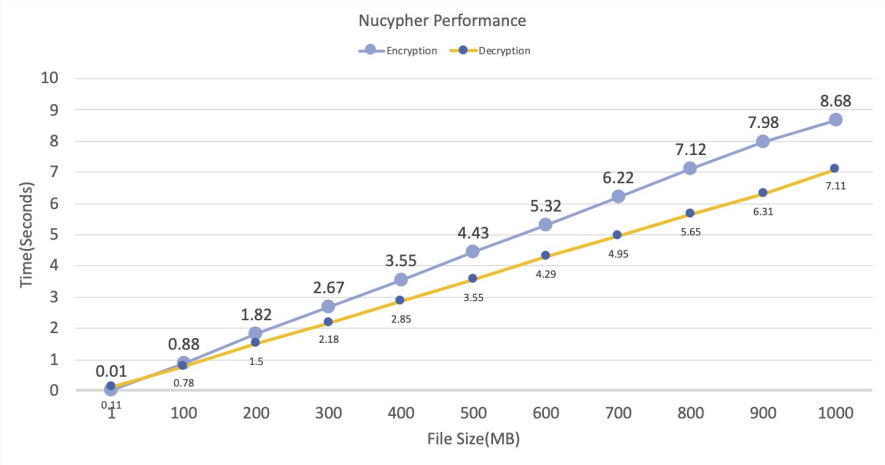


Figure 4.15: Nucypher Encryption and Decryption Performance

Chapter 5

CONCLUSION

5.1 Performance Summary

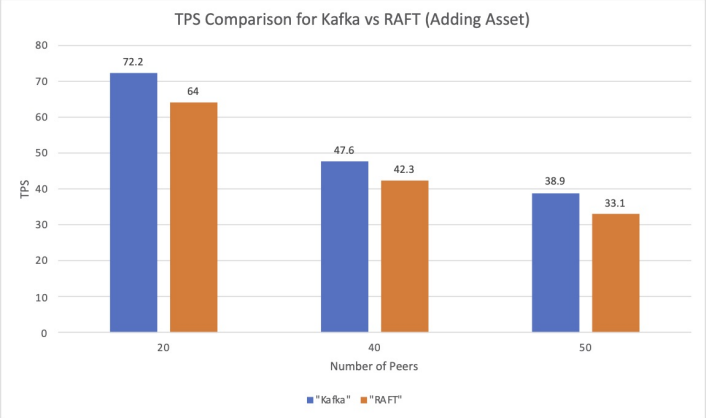


Figure 5.1: Kafka vs RAFT (Adding Asset)

Summarizing all performance tests for adding assets onto the blockchain, we observed:

- Throughput keeps decreasing as we increase peers. As a decentralised blockchain application, Increasing peers means increasing endorsement, more ledgers and more performance strain. Performance almost halved when we increased peers from 20 to 50. This analysis shows, although HLF is scalable but is limited to permissioned, business centric solutions only.

- Kafka outperformed RAFT every time, although very closely. This may be due to the requirement of TLS for operating a RAFT node.

- For both, Kafka and RAFT, high throughput and high batch size observed more transaction failures in higher peer networks. This is due to the Byzantine Fault tolerant network. higher peer network needs more time to process a transaction which causes failure under strain.

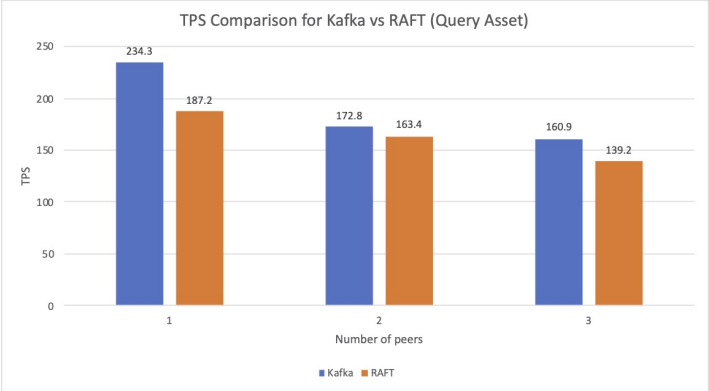


Figure 5.2: Kafka vs RAFT (Adding Asset)

Summarizing all performance tests for adding assets onto the blockchain, we observed:

- Query throughput keeps decreasing as well with increase in peers but, drop in transaction rates are significantly lower than adding asset drop rates. This is due to the fact that are no endorsement policies for querying chaincode.
- Kafka again outperformed RAFT.

Other than HLF performance benchmark, Nucypher and IPFS proved to be scalable with their own set of benefits.

- Nucypher performed encryption and decryption quite fast and in linearly increasing order. It took nucypher less than 10 seconds to encrypt and decrypt 1Gb data without compromising data itself.

- IPFS uploading speed heavily depends upon whether data is hashable or not. For non-hashable data, records are instantly available. While, hashing may take more time, it provides unique hash value that further ensures data integrity.

5.2 HIE vs MedFabric4Me

MedFabric4Me experiment has shown potential in an interoperable network of private and public blockchain. If we compare this application with currently HIE, interms of scalability and performance:

- **Privacy:** As we have discussed earlier, healthcare and EHR data is protected by federal and state laws. Thus every system should be compliant by HIPAA privacy guidelines. MedFabric4Me satisfies all minimum requirements.
  - A patient can delete their account anytime on MedFabric4Me, thus opting out of network and MedFabric4Me immediately deletes all individually identified information for the participant. Although, any exchange of information should be recorded as a transaction history on MedFabric4Me as ledger is immutable.
  - A patient controls sharing of data in MedFabric4Me, thus can opt-out anytime from sharing records. A patient can share policies of already shared data as well.
  - A patient can ask provider for amendment in incorrect health information. Although, patient cannot change EHR themselves, any discrepancy can be amended by following doctor.
- **Partial Data Sharing:** De-identification is required when patient's data is shared for research purposes. To comply with the same, MedFabric4Me can share partial data, hiding personal information or not sharing non-required

data for structured data only. For unstructured data, it is not implemented in the system but could be done at later stage.

- **Scalability:** For scalability of HLF, we will compare it with state of art Healthcurrent stastistics. Healthcurrent performs average 23.4M HL7 transactions, 1.2M CCD transactions and 10.4M alerts per month. In comaprison, we tested our network for different number of peers and results are as follows :

Table 5.1: Per Month Records(Adding Assets) According to TPS

Number of peers	Kafka (in millions)	RAFT (in millions)
20	187	166
40	123	110
50	101	86

- **IPFS vs Cloud:** Cloud services provides approximately 20Mb/s in upload and download speed[23]. While IPFS provides a consistent 0.4 seconds upload time for encrypted files while for non-encrypted files, speed decreases with increase in file size as shown in figure below.

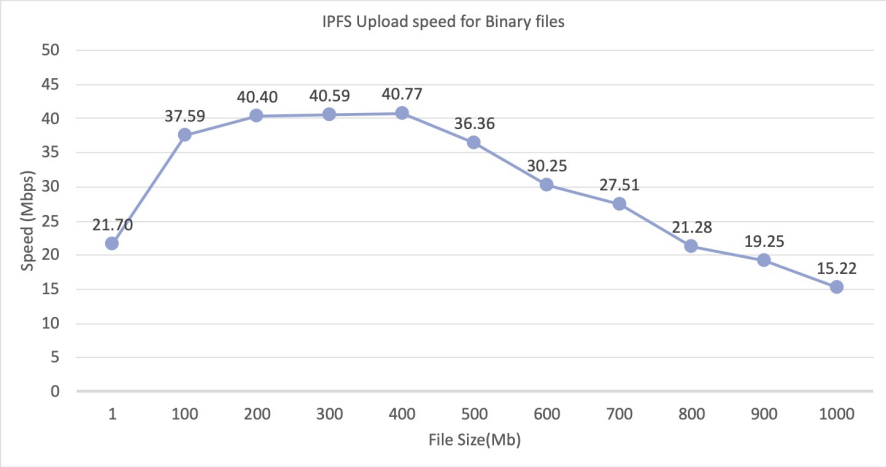


Figure 5.3: IPFS Uploading Speed

This shows that IPFS performs better in uploading than cloud services for most non-encrypted file sizes as well.

In case of Downloading, IPFS provides an approximately 7Mbps speed which is lower than downloading speed of cloud services. This may be improved by creating a private IPFS cloud service and duplicating data over multiple peers.

We can conclude that IPFS may not be the best database for application but it serves the purpose of saving file by hash and having access management through HLF.

- **Nucypher:** The most popular cloud based encryption algorithm provides around 55MB/s speed for encryption[9], when compared Nucypher provides 115.20MB/s.

  PRE provides better performance in terms of encryption time than existing cloud services.

5.2.1 Use Case Comparison Between MedFabric4Me and HIE

Let's take some use case scenarios where HIE provides benefits to healthcare providers and participants, and compare them with implementation of MedFabric4Me.

1. **Admission, Discharge, Transfer (ADT) messages:**In an HIE, when a patient is admitted, discharged or transferred to an emergency setting, providers caring for the patient received an alert with basic information (date admission, place of admission, the reason for admission, etc) and can reach the patient to follow up.

   MedFabric4Me does not have an emergency visit implementation yet, although the main functionality of ADT messages is implemented in an immutable ledger

way. Whenever a patient visits an emergency and records gets updated, MedFabric4Me makes an immutable record of it in the transaction. Providers do not get any ADT messages at the exact time of status change, but whenever a patient visits providers, they get an overall history of record update and view when the patient visited the emergency and for what reasons.

2. **Pandemic Alert Enhancements:** A pandemic is an epidemic of disease that has spread across a large region, for e.g SARS, COVID-19. Generally, during a pandemic, multiple institutes and testing labs provide quick results to the mass population. HIE needs to adapt and create an infrastructure to adapt to a pandemic, that initially slows response. Even once the infrastructure is in place, security is always vulnerable as alerts need to be real-time and at large scale. For example, during COVID-19, Healthcurrent initially provided alerts and data without the Secure File Transfer Protocol.

   On the other hand, MedFabric4Me does not need to adapt according to the pandemic each time. Although, MedFabric4Me does not provide emergency based alerts, we do not need to create a new functionality or complete infrastructure to support pandemic reporting. An institute or testing lab that is doing mass testing can securely provide results to the respective patient, which then further exchanges information with different providers on a need-to-know basis. Similarly, testing results can also be shared with different research institutes who require verified and trusted data to do research. As a testing lab is just another provider in our application, they can securely use already established security protocols for record sharing.

3. **Medical Device Management:** HIE has very specific functionality, which is to share medical health records among different providers and participants. The

medical industry is vast and involves multiple other industries. One of them is medical device manufacturers who provide medical devices to providers, research institutes and healthcare facilities. Manufacturers charge providers or institutes based upon the use of the particular device. Currently, medical manufacturers have limited visibility over device usage, which results in inefficient service.

MedFabric4Me's partial data sharing policies can help device manufacturers to have proper visibility about device usage among providers while maintaining data confidentiality. The manufacturer can find out which medical devices need software updates and reach out to particular facilities/providers for services. Other than that, manufacturers can have visibility into inventory at a particular location, which helps them to make a thoughtful decision whether to move that machine within different providers at the location or need to call for a new machine.

With above experimentation and comparison we can conclude that MedFabric4Me is a viable decentralised alternative to currently present centralised alternative. Some outlined benefits of MedFabric4me will be :

- MedFabric4Me has the capacity to perform better in respect to Throughput.
- Decentralisation also makes sure no single point of failure.
- Reduces overhead cost of cloud computing, costly encryption and helps transparent cost distribution between insurers, providers and patients.
- MedFabric4Me provides data governance to Patients, thus increasing transparency, decreasing delay and improving services.

Although MedFabric4Me showed potential in some areas, there are some areas where it lacked to currently present centralised solution:

- **Ease of Setup:** It is very easy to setup and scale a centralised system than a decentralised system like MedFabric4Me. Centralised HIE is mostly built on enterprise cloud computing, which offers multiple services to manage, scale and analyse system applications. For MedFabric4Me, it is complicated to create a enterprise level system application with multiple independent components.

- **System Upgrade and Updates:** It is easier to update a centralised systems, as all servers and application codebase is controlled from one central location. Blockchain setups are hard to upgrade as nodes are distributed and each node to update simultaneously for keeping the network functional.

- **Easier Data Backup:** As a centralised system, it is easier for current HIE to backup health data. This could be complex for MedFabric4Me as patients own their data and are at liberty to choose their own service providers for data storage. It may be possible that patient can loose data permanently due to lack of data backups.

5.3 Future Work

Some of the shortcomings of current MedFabric4Me could be addressed in further iterations. Some of the ways to further improve MedFabric4Me could be :

- **Emergency Response:** Current implementation does not support emergency cases where patient is enable to respond or provide data sharing permissions. As we shown multiple ways to share data among different actors, this emergency response can be implemented by encrypting crucial healthcare data in

advance and delegate its access to emergency contact. Once records are used in emergency, policy will expire and you need to create new policy with new pair of encryption keys.

- **Enterprise HLF deployment services:** Multiple cloud services provide enterprise level support for HLF. If we could create or choose a deployment service for HLF then it becomes easier to maintain a large network and onboard new peers.

- **Improving Data Storage:** We have seen from experiments that IPFS performs poorly in comparison with cloud computing as a data storage service. We should switch to a database service with better performance and data backup solutions.

[1] "Architecture", URL https://hyperledger.github.io/caliper/vNext/architecture/.

[2] "Demystifying hyperledger fabric architecture", URL https://www.serial-coder.com/post/demystifying-hyperledger-fabric-architecture/.

[3] "Health current", URL https://healthcurrent.org/?page=Network_Patients.

[4] "Hyperledger caliper", URL https://hyperledger.github.io/caliper/.

[5] "Merkle trees", URL https://hackernoon.com/merkle-trees-181cb4bc30b4.

[6] "Electronic health record", URL https://en.wikipedia.org/wiki/Electronic_health_record.

[7] "Health information exchange", URL https://en.m.wikipedia.org/wiki/Health_information_exchange.

[8] "Merkle tree", URL https://en.wikipedia.org/wiki/Merkle_tree.

[9] Aminubaba, M., A. Yusuf, A. Ahmad and L. MaijamaA, "Performance analysis of the encryption algorithms as solution to cloud database security", International Journal of Computer Applications **99**, 14, 24–31.

[10] Androulaki, E., Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou and et al., "Hyperledger fabric", Proceedings of the Thirteenth EuroSys Conference on - EuroSys 18 .

[11] Azaria, A., A. Ekblaw, T. Vieira and A. Lippman, "Medrec: Using blockchain for medical data access and permission management: Semantic scholar", URL https://www.semanticscholar.org/paper/MedRec:-Using-Blockchain-for-Medical-Data-Access-Azaria-Ekblaw/bd8a307efcffbf57d2e5c3c23577de44d883d865.

[12] Benet and Juan, "Ipfs - content addressed, versioned, p2p file system", URL https://arxiv.org/abs/1407.3561v1.

[13] Dagher, G. G., J. Mohler, M. Milojkovic and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology", URL https://www.sciencedirect.com/science/article/abs/pii/S2210670717310685.

[14] Dubovitskaya, A., Z. Xu, S. Ryu, M. Schumacher and F. Wang, "Secure and trustable electronic medical records sharing using blockchain", URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977675/.

[15] Egorov, Michael, Wilkison, MacLane, Nunez and David, "Nucypher kms: Decentralized key management system", URL https://arxiv.org/abs/1707.06140.

[16] Good, T., "2017 healthcare trends part 3: The rise and fall of the ehr", URL https://datica.com/blog/2017-healthcare-trends-part-three-the-rise-and-fall-of-the-ehr/.

[17] Kirsh, D., W. K. and W. K., "How outdated medical record systems and devices could risk lives", URL https://www.medicaldesignandoutsourcing.com/outdated-medical-record-systems-devices-risk-lives/.

[18] Llp, B., "The financial cost of healthcare fraud 2014", URL https://www.prnewswire.com/news-releases/the-financial-cost-of-healthcare-fraud-2014-252162971.html.

[19] Mamun, M., "How does hyperledger fabric work?", URL https://medium.com/coinmonks/how-does-hyperledger-fabric-works-cdb68e6066f5.

[20] Ocr, "2057-what is the intersection of the hipaa right of access and the hitech act's medicare and medicaid electronic health record incentive program's "view, download, and transmit" provisions?", URL https://www.hhs.gov/hipaa/for-professionals/faq/2057/what-is-the-intersection-of-the-hipaa-right/index.html.

[21] of the National Coordinator for Health Information Technology, O., "Report to congress: Report on health information blocking", .

[22] Swan, M., "Blockchain - blueprint for a new economy - pdf free download", URL https://epdf.pub/blockchain-blueprint-for-a-new-economy.html.

[23] Taylor, J., "Cloud storage comparison: Top 4 fastest cloud storage", URL https://www.imobie.com/cloud-manager/fastest-cloud-storage.htm.

[24] Team, D., "Types of blockchains - decide which one is better for your investment needs", URL https://data-flair.training/blogs/types-of-blockchain/.