Design and Implementation of Efficient Personal Health-Records Sharing in IoMT using Searchable Symmetric Encryption, Blockchain and IPFS

Thesis submitted in partial fulfillment
of the requirements for the degree of

*(Master of Science in **Computer Science and Engineering** by Research)*

by

ABHISHEK BISHT
2021202014
abhishek.bisht@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Design and Implementation of Efficient Personal Health-Records Sharing in IoMT using Searchable Symmetric Encryption, Blockchain and IPFS" by Abhishek Bisht, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Dr. Ashok Kumar Das

Dedicated to my family

Acknowledgments

The completion of this MS thesis would not have been possible without people who guided and supported me directly and indirectly throughout my MS journey. I take this opportunity to thank and acknowledge them sincerely.

I am deeply grateful to my MS advisor, Dr. Ashok Kumar Das, who guided and supported me by providing timely and valuable feedback on my work. With his constant guidance and support, the completion of this thesis became possible.

Next, I would like to thank my parents and family for their unconditional support throughout the duration of my MS program. Without them, this accomplishment would not have been possible.

I am also thankful for the support from the senior students under Dr. Ashok Kumar Das and the Center for Security Theory and Algorithmic Research Teaching staff, who helped me complete my thesis work.

Lastly, I am sincerely grateful to all the non-teaching staff of the International Institute of Information Technology, Hyderabad, especially the staff of Kadamba Niwas and Kadamba Mess, for their support.

Place : IIIT Hyderabad

Date:

Abhishek Bisht
MS Student
Roll No. 2021202014
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, INDIA

Abstract

In this thesis, we focus on the secure storage and sharing of Personal Health Records (PHRs) in Internet of Medical Things (IoMT). Due to the high value of personal health information, PHRs are one of the favorite targets of cyber attackers worldwide. Over the years, many solutions, including those based on blockchain, have been proposed; however, most solutions are inefficient for practical applications. For instance, several existing schemes rely on bilinear pairings, which incur high computational costs. To mitigate these issues, we propose a novel PHR-sharing scheme that is dynamic, efficient, and practical. Specifically, we combine searchable symmetric encryption, blockchain technology, and a decentralized storage system known as the Inter-Planetary File System (IPFS) to guarantee the confidentiality of PHRs, verifiability of search results, and forward security. We start by introducing cloud and blockchain and then provide a detailed literature survey of existing works that make use of these technologies for storage and sharing of health records. We then propose our own scheme which provides better security and better efficiency than existing schemes for storage and sharing of PHRs. Moreover, we provide formal security proofs for the proposed scheme and also provide extensive test-bed experimental results which demonstrate that the proposed scheme can be used in practical scenarios related to the IoMT environment.

Keywords: Healthcare system, Searchable encryption, Blockchain, Cloud computing, Personal Health Records (PHRs), Secure storage, Security, Internet of Medical Things (IoMT), Inter-Planetary File System (IPFS) decentralized storage.

Dissemination of Work

Chapter #3.

Abhishek Bisht, Ashok Kumar Das, and Debasis Giri. "Personal Health Record Storage and Sharing using Searchable Encryption and Blockchain: A Comprehensive Survey," in ***Security and Privacy (Wiley)***, 2023, DOI: 10.1002/spy2.351. (2022 SCI Impact Factor: 1.9)

Chapter #4.

Abhishek Bisht, Ashok Kumar Das, Dusit Niyato, and YoungHo Park. "Efficient Personal-Health-Records Sharing in Internet of Medical Things using Searchable Symmetric Encryption, Blockchain and IPFS," in ***IEEE Open Journal of the Communications Society***, Vol. 4, pp. 2225-2244, 2023, DOI: 10.1109/OJCOMS.2023.3316922. (2022 SCI Impact Factor: 7.9)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Blockchain has been one of the most popular technologies over the past decade and has found applications in broad areas far from its first application - cryptocurrency. One such area is searchable encryption, which aims to provide an efficient and secure encryption mechanism to allow authorized users to search over the data encrypted by the data owner and stored on a remote storage such as the cloud. Searchable encryption has various applications, one of which is tied to health care. Hospitals prefer to store patients' health records digitally because it allows efficient management and quick information retrieval. However, maintaining local storage for high-volume data is a non-trivial and expensive task. Hence, outsourcing the data to third-party storage providers is the preferred way to store the records. But again, outsourcing the records as plain text to some storage service provider is a direct breach of privacy as the third-party service provider cannot be trusted. Therefore, health records are typically encrypted and then stored on remote storage at the expense of efficient information retrieval. A significant advantage of electronic health records is lost due to a lack of query functionality over encrypted data. This is where searchable encryption comes into play, which lets users query encrypted data securely.

Moreover, the cloud, a traditional storage medium, has a disadvantage in that the cloud server needs to be trusted against any dishonesty, such as returning false data for cost savings. To counteract this issue, various existing works [5, 6, 7, 8], have incorporated blockchain as a mechanism to detect dishonest behavior by cloud service providers. In some instances, the entire cloud server has been replaced by a distributed blockchain network, storing health records as transactions. In the blockchain environment, transactions are immutable, and the validity of any block can be easily verified. This characteristic

ensures that health records stored in the form of transactions on the blockchain network remain tamper-proof, addressing concerns related to data integrity and security.

In this chapter, we describe the various technologies mentioned above in detail, so that the subsequent discussion of the application of SE and blockchain in healthcare become easier to understand.

1.1 Cloud computing

Cloud computing is the practice of delivering computing resources such as computing, storage, databases, network and analytics through a pay-as-you-go model over the internet without managing or setting up physical infrastructure [9]. It is an on-demand provision of services by cloud service providers, such as "Amazon Web Service (AWS)", "Microsoft Azure", "Google Cloud Platform", and "IBM Cloud Services" to the users with a service level agreement(SLA) to ensure robust delivery of resources.

According to the "National Institute of Standards and Technology (NIST), USA", cloud computing can be defined as follows [10]:

> "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

This cloud model consists of five necessary characteristics - on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service; four deployment models - public cloud, private cloud, community cloud and hybrid cloud; and three service models - Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

Cloud Computing is a cost-effective and reliable solution for those who want to reduce the hassle of physically setting up the whole infrastructure. It is highly scalable within no time. Based on consumption, users can increase the resources within a couple of minutes. Moreover, with cloud resources, security is also taken care of by *Cloud Service Providers (CSPs)*.

Following are the characteristics of cloud computing listed in detail.

1.1.1 On-demand services

Cloud on-demand service typically refers to cloud computing services available for immediate usage as needed without requiring a long-term commitment or upfront investment. In other words, it is a pay-as-you- go model for cloud resources and services. Cloud on-demand services allow businesses and individuals to access a wide range of computing resources, including virtual machines, storage, databases, and software applications, without purchasing and maintaining their physical infrastructure. Users can scale their resources up or down as their needs change and are only charged for the resources they use. These flexibility and scalability make cloud on-demand services attractive for various purposes, such as hosting websites, running software applications, storing and managing data, and more. Leading cloud service providers like Amazon Web Services (AWS), Microsoft Azure and Google Cloud offer on-demand services that have become essential for modern computing and IT infrastructure.

1.1.2 Scalability and elasticity

Scalability and elasticity are two critical features in cloud computing that allow businesses and organizations to effectively manage their computing resources based on demand and optimize performance, cost-efficiency, and user experience.

- *Scalability:* Scalability refers to the ability of a system, application, or service to handle an increasing amount of work or traffic. In cloud computing, scalability is typically categorized into two types:
  - Vertical scaling - Increasing the capacity of a single machine or resource, such as adding more CPU, RAM, or storage to an existing server. However, this has limitations and can become costly.
  - Horizontal scaling - It involves adding more machines or nodes to a system, effectively distributing the workload across a cluster of smaller, interconnected devices. This approach is more flexible and cost-effective for handling larger workloads.

  Cloud platforms enable both types of scalability. Horizontal scalability is fundamental, allowing applications to scale dynamically by adding or removing instances as demand fluctuates. It ensures the application remains responsive and performs well under varying usage levels.

- *Elasticity:* Elasticity is an extension of scalability and refers to the ability to automatically provision and de-provision resources based on actual usage requirements. It ensures that the system can adapt to the real-time workload changes, providing optimal performance and cost-efficiency. Critical aspects of elasticity include the following:

  - *Automatic scaling:* The ability to automatically adjust resources based on predefined policies, such as scaling up during peak demand and down during off-peak periods.
  - *Cost management:* Elasticity helps optimize cost by allocating resources as needed, preventing over-provisioning, and minimizing unnecessary expenses.
  - *Resource management:* Cloud platforms can monitor usage patterns and predict when additional resources are needed, scaling up in anticipation of increased demand.

  Elasticity is essential for applications and services with unpredictable or fluctuating workloads, because it ensures that the required resources are available and can be released when no longer necessary. This approach helps organizations optimize their cloud spend and maintain high-performance levels even during traffic spikes.

1.1.3 Multi tenancy

In cloud computing, Multi-tenancy refers to a model where multiple customers, called tenants, share the same infrastructure and computing resources within a cloud environment. These tenants could be individual users, businesses, or organizations, each with their applications, data, and virtual instances running on a shared pool of resources. Multi-tenancy is a core feature of public cloud services that major cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) provide. These providers offer services that allow multiple customers to use shared resources while maintaining robust security and tenant isolation. However, it requires robust security and isolation measures. Security controls, such as encryption, access control, and network segmentation, are critical to maintaining the privacy and security of tenant data. The following are some advantages of multi-tenancy:

- *Cost-efficiency:* Multiple tenants can benefit from economies of scale by sharing the

same underlying infrastructure. This reduces hardware, maintenance, and management costs since the expenses are spread across multiple users.

- *Scalability:* Multi-tenancy allows for easy scaling of resources, accommodating customers' changing needs without significant lead times or resource provisioning.

- *Flexibility:* Tenants can provision and manage their own virtual instances, storage, and applications, with the ability to customize their environments while still sharing the underlying infrastructure.

- *Resource isolation:* While tenants share resources, cloud providers implement robust isolation mechanisms to ensure customer data security. It helps in preventing data leakage and maintain privacy.

1.1.4 High availability

In cloud computing, high availability refers to the design and implementation of systems, applications, and infrastructure in a way that minimizes downtime and ensures continuous, reliable access to services and resources. It aims to keep services up and running even in the face of hardware failures, software issues, network problems, or other disruptions. It is critical for mission-critical applications and services, and cloud computing provides the infrastructure and tools needed to achieve it. However, achieving it requires careful architecture, redundancy, regular testing, and ongoing monitoring and maintenance to ensure that services remain resilient and accessible. Cloud computing provides a robust platform for achieving high availability through various strategies and practices listed as follows:

- *Redundancy:* Redundancy involves duplicating critical components (such as servers, storage, and networking) to ensure that if one component fails, another can take over seamlessly. Cloud environments offer load balancers and auto-scaling groups that distribute traffic and workloads across multiple instances or data centers.

- *Load balancing:* Load balancers distribute incoming network traffic or requests across multiple servers or resources to prevent overloading any single component. It enhances performance and ensures that others can handle the traffic if one server fails.

- *Auto-scaling:* Auto-scaling allows cloud resources to adjust to changing workloads automatically. Additional instances are provisioned for increased demands, and extra instances are terminated when demand decreases. It helps maintain consistent performance during traffic spikes and minimizes costs during quiet periods.

- *Geographic distribution:* Cloud providers can replicate resources and data across multiple geographic regions. This geographic redundancy ensures that services can be seamlessly shifted to another location if a data center or area experiences an outage.

- *Fault tolerance:* Cloud services and infrastructure are designed with fault tolerance in mind. It means that even if a component fails, there are mechanisms to ensure service continuity without manual intervention.

Cloud computing can be divided into the following models based on the service provided by them.

1.1.5 Infrastructure as a Service (IAAS)

IAAS, also known as Hardware as a Service, is the provision of cloud computing infrastructures like servers, storage, operating systems, and networks as a service. These services are available to users based on demand and can be scaled whenever required based on load. Through this service model, companies can buy high processing power CPUs, memory, etc, without spending much time and capital expenditure on setting up physical servers and infrastructure. Some of the IAAS offering examples are "Rackspace, Amazon Web Services (AWS) DigitalOcean, Cisco Metapod, Google Compute Engine (GCE), Linode, and Microsoft Azure".

1.1.6 Platform as a Service (PAAS)

PAAS delivers cloud platform services, i.e., a runtime environment over which applications can be built, tested, and deployed. PAAS services provide a framework for developers to develop their software further. These services include both Infrastructure as a Service (IAAS) and Platform as a service (PAAS). Using PAAS offerings, developers can directly concentrate on building and designing applications without putting much effort into runtime environment maintenance, software updates, underlined operating systems,

and infrastructure. PAAS providers, such as "Azure, Google App Engine, Force.com, etc." provide application frameworks, databases, programming languages, etc.

1.1.7 Software as a Service (SAAS)

SAAS, also known as "On Demand Software," offers cloud applications services which include end-to-end ready software to use. In these services, cloud providers provide infrastructure, platforms, and software and maintain all technical aspects of the offering, such as data, middleware, storage, and servers. Through SAAS services, users can reduce the time spent on installing, updating, and maintaining software on individual systems. Rather, everything can be accessed and managed through a central location on the cloud. SAAS providers, such as Salesforce, Netflix, Google Workspace apps, etc., provide offerings like cloud based mail services, social networks, and collaborative document sharing.

1.2 Blockchain

Blockchain was first introduced in Satoshi Nakamoto's white paper [11] as the technology upon which Bitcoin (the first cryptocurrency) is based. Following that, the researchers started exploring the applications of blockchain in other areas. In recent years, blockchain has found application in many areas, such as healthcare applications [12, 13, 14, 15, 16, 17], crowdsourcing system [18], Industrial Cyber-Physical Systems (ICPS) [19], Internet of Drones (IoD) [20, 21, 22, 23, 24], Internet of Things (IoT)-enabled smart agriculture [25, 26, 4], Internet of Vehicles (IoV) and Vehicular Adhoc Network (VANET) [27, 28, 29, 30], Software-Defined Network (SDN) [31, 32], Industrial Internet of Things (IIoT) [33, 34], Internet of Everything (IoE) [35], smart-grid system [36], supply chains [37], Cognitive Internet of Things (CIoT) [38, 39, 40], and so on. The various applications of blockchain technology are also illustrated in Figure 1.1.

Blockchain acts as a distributed ledger that records immutable transactions. However, it is not limited to financial transactions as in Bitcoin. It can record any other type of data that can be represented in the form of a transaction, such as the occurrence of an event of any sort. The primary benefit of blockchain is that once a transaction has been recorded, it is immutable. To nullify the effect of a previous transaction, one will have to create a new transaction. These transactions are stored in the form of a block where a single block may contain an arbitrary number of transactions depending on the

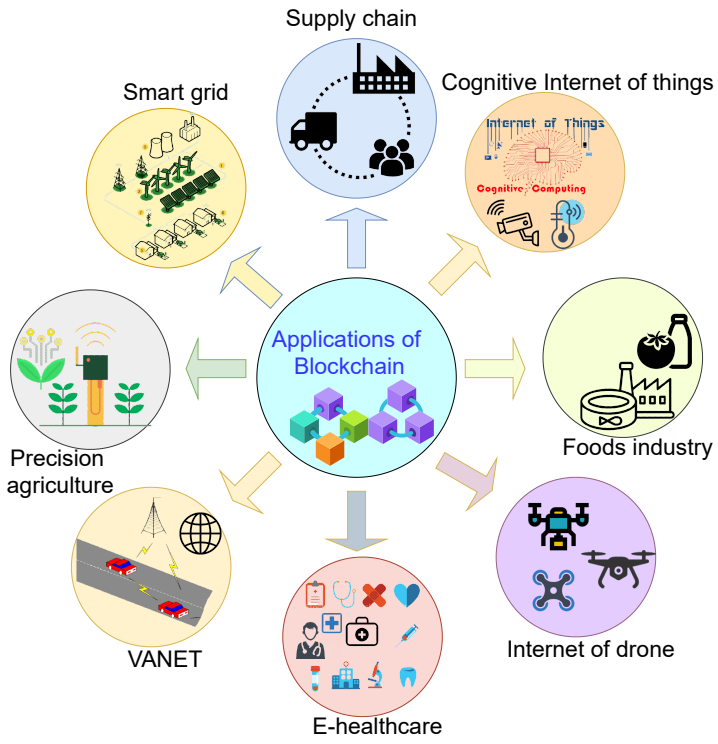

Figure 1.1: Various applications of blockchain (Adopted from [1])

requirement and context in which the blockchain is being used. A blockchain network contains many independent nodes which interact with each other and make decisions based on an agreement algorithm, known as a consensus algorithm.

A block is nothing but binary data which is divided into two parts: 1) *header* and 2) *payload*. The header stores meta-data related to the block, one of which is the hash of the previous block, and the payload contains the data specific to a transaction. Since each block (except the first one) has the hash of its predecessor, all the blocks together form a chain, and hence, the data structure got its name as blockchain. The first block, which does not have any predecessor, is known as *genesis* block. Each node participating in the blockchain network has its copy of the blocks. The longest chain in the network is considered to be the valid chain by all the nodes. Any modification to a partnership requires that all the blocks ahead of it in the chain be modified. A malicious entity will have to alter the contents of the majority of the nodes in the network by breaching them, which is almost impossible to do if independent entities control the nodes. Thus, when a malicious entity tries to alter the contents stored on a block, it gets detected very easily and other nodes involved in the maintenance of the blockchain can take appropriate

action.

When a node wants to add a new block to the chain, it broadcasts this block to all the nodes in the network and each node verifies the integrity of this block before adding it to its local copy. Further, to decide upon the node that will be allowed to add the new block in the network, the consensus algorithm is used.

1.2.1 Consensus algorithms

Consensus is one of the fundamental problems associated with distributed systems. Many of the algorithms used in distributed systems rely on the consensus algorithm for their correctness. Faulty nodes can also be of two types - crash faulty and byzantine faulty. Crash faulty processes can crash in real-time. However, faulty Byzantine processes can behave as a malicious entity that tries to prevent other processes from reaching a consensus.

In the following, we list some popular consensus algorithms.

- *Proof of Work (PoW):* In this algorithm [41], a hard problem is selected and given to every node. The node that solves this problem earliest gets the chance to add a new block to the chain. This algorithm is used in Bitcoin. However, it is very energy inefficient, and hence, alternative solutions to it are being considered now.

- *Practical Byzantine Fault Tolerance (PBFT):* This algorithm [42] guarantees that a consensus is reached if the number of faulty nodes is at most $\frac{n-1}{3}$, where n is the total number of nodes in the blockchain network. It is an energy-efficient algorithm. However, it does not scale well to a large number of nodes, as the communication overhead is relatively high.

- *Proof of Elapsed Time (PoET):* This algorithm was developed by Intel Corp. It follows a lottery system in which each node randomly sleeps for a random period called wait time. The node with the smallest wait time wins the lottery and gets a chance to add a new block to the chain. This algorithm relies on secure computing, as the sleep algorithm must run in a tamper-proof way so that a malicious node does not alter the randomness of sleep time for its benefit.

- *Raft:* Raft [43] is equivalent to an earlier class of algorithms called Paxos [44] but is easier to understand and is proven to be safe, unlike Paxos. The algorithm selects a leader among the nodes in the network, and the leader has a tenure after which

elections for a new leader are held. Once a leader is elected, it is responsible for the coordination of all the nodes.

- *Ripple Protocol Consensus Algorithm (RCPA):* In the network running Ripple algorithm [45], each server/node maintains a Unique Node List (UNL), which consists of the nodes that this node queries when determining consensus. Each node runs the Ripple algorithm every few seconds, and if 80% of a node's UNL agrees on a transaction, then it is applied.

- *Proof of Stake (PoS):* Under this algorithm [46], each node stakes some amount of cryptocurrency to become a candidate for the validator in the network. An algorithm then selects one node out of the candidate set based on a combination of the amount of cryptocurrency staked along with some other factors. It assigns it as the validator for the next block. The validator then validates the next block and adds it to the network in exchange for a fee.

- *Proof of Vote (PoV):* This algorithm [47] is used in case of consortium blockchain 1.2.2. There are four major roles in this algorithm - "Commissioner", "Butler", "Butler-candidate" and "Ordinary User". The commissioner is part of the consortium committee of the network and verifies each block that is generated in the network. Butlers specialize in the production of blocks and are elected by the commissioner for a specific tenure , which expires after a certain period. The next role - butler-candidate, participates in the election for becoming butler. Ordinary users, on the other hand, can join or exit the network anytime without any authorization.

A comparative study among various blockchain consensus protocols is summarized in Table 1.1.

1.2.2 Types of blockchain

We can categorize blockchain into three types based on the relation between the nodes participating in the blockchain network.

- **Private blockchain:** In this type of blockchain, all the nodes belong to a single organization, and any external node cannot join the network without getting permission from the organization. An example of private blockchain includes the healthcare [54].

Table 1.1: Comparative study among various blockchain consensus protocols [4]

Consensus Protocol	Concept Applied	Resource Used	Application Areas
PoW	Hashing	Computations	Ethereum [48] Bitcoin [11]
PoS	Digital signatures	Currency	SnowWhite [49] PeerCoin [50] Ouroboros [51]
PBFT	Voting	No resource	Tendermint [52]
Ripple	Voting occur in multiple rounds	No resource	XRP ledger [53]
PoET	Generation of random numbers	Intel SGX	Hyperledger Sawtooth [2]

- **Public blockchain:** In Public blockchain, the nodes can belong to different individuals and organizations. The most popular blockchain network - Bitcoin, is an example of a public blockchain [11].

- **Consortium blockchain:** Consortium blockchain is a mixture of the first two types. It consists of a core set of nodes that have exclusive rights to validate a new transaction and add a new block to the chain. All other participants can only submit transactions and necessarily participate in block forwarding. An example of consortium blockchain includes Intelligent Transportation System (ITS) [55].

1.2.3 Smart contract

As described earlier, blockchain stores data in the form of transactions that are immutable once committed. This data need not be passive. Instead, it can also be active, such as some piece of code. Because of the immutable property of a transaction, the code cannot be modified by any malicious entity unless it controls the majority of the nodes in the blockchain. Further, this code runs only when certain conditions are met. These conditions are set by the contract's creator and embedded in the code itself. Because of smart contracts, blockchain can be used for various applications. Ethereum originally introduced smart contracts [56], which is a public blockchain platform. Now, there are even alternatives to smart contracts that are more efficient and flexible. One of those is the Hyperledger Sawtooth, which uses transaction processors instead of smart contracts to achieve the same functionality.

1.2.4 Advantages of blockchain

Blockchain has several advantages over cloud that are listed below.

- *Transparency:* All the nodes participating in the blockchain network hold a copy of the blocks, making the transactions transparent to all the nodes.
- *Security:* Once a transaction is committed to a block, it cannot be mutated by anyone, i.e., immutable. This blockchain property ensures that the data stored in the blocks is secure from any malicious entity.
- *Decentralization:* Blockchain is a distributed network in which each participating node has equal power. There is no central authority; hence, a single point of failure is eliminated.
- *Traceability:* All the blocks are linked in the form of a chain, which makes it very easy to trace the transactions that have occurred up to any depth. This property of blockchain, which makes it very easy to trace previous transactions, has applications in many areas, a major one of which is supply chain management [57].
- *Immutability:* The immutable property is achieved on the transactions that need to be agreed upon as well as shared across the blockchain network. Now, if the transactions are connected to the blockchain, it becomes difficult to modify or erase them [58].

1.3 Searchable encryption

With the ubiquitous presence of cloud storage, in order to mitigate the risk of data leakage, one would prefer to encrypt the data and then upload it to a cloud server. However, this results in the loss of many desirable functionalities, out of which a prominent one is the search for the stored data. Searchable Encryption (SE) mitigates this loss by encrypting data in such a way that one can search for keywords in the stored data without actually decrypting it. Research in the SE domain was pioneered by Song *et al.* [59] in the year 2000. They proposed the first searchable encryption scheme, which used a linear search technique in which the entire encrypted document needs to be processed during the search. However, it could be more efficient when the size of document and document sets becomes large. A second approach was suggested in the work by Song *et al.*, and later

used by many subsequent works, which creates an index over the encrypted files that are then used during the search to look for the files where this keyword occurs. The latter approach is significantly faster than the former one and was therefore studied in more detail later on by other researchers. The original linear scan technique by Song *et al.*, being very naive and inefficient, is not considered for further study. Later, other forms of searchable encryption, such as homomorphic encryption-based searchable encryption, fuzzy keyword-based search and public key encryption using conjunctive keyword search were also introduced. However, index-based searchable encryption remains the most efficient one, and much work has been done in this direction.

SE can be widely categorized as *static* and *dynamic*. *Static* SE refers to the case where the files are encrypted as a batch, and after encryption, files cannot be added or deleted without requiring a re-encryption of the modified set of files. *Dynamic SE*, on the other hand, allows efficient addition and deletion of files without needing a re-encryption. SE can also be categorized as *symmetric* or *asymmetric* based on the encryption type that lies at its core. Moreover, depending on the number of keywords that can searched, SE is categorized as *single-keyword* or *multi-keyword*. Multi-keyword search is more desirable; however, it is challenging to construct schemes supporting it without leaking more information.

1.4 Personal health records

A Personal Health Record (PHR), is a digitized form of a patient's medical record that contains the entire medical history of the patient and is maintained by the patient. Ideally, a patient can grant access to other users, such as medical institutions and insurance companies, as per his or her requirements. PHRs make it easier to share and access the medical history of a patient in an efficient manner that saves time and cost for both the patient and the medical institution. In an emergency, PHRs can be very helpful as the previous medical history is often needed before giving any treatment. People often get confused with "Electronic Medical Records (EMRs)" and "Personal Health Records (PHRs)". PHR and EMR differ in the sense that the patient himself maintains the former, and the latter is created and maintained by the hospital or medical institute. However, the difference between PHR and EMR can be subtle. It can, however, be clarified by considering the difference between the terms 'medical' and 'health'. An EMR is a narrow view of a patient's medical history created by a doctor to diagnose the current disease.

On the other hand, a PHR is a comprehensive medical history of the patient related to the diagnosis of current conditions and past ailments. However, the two terms are often used interchangeably.

1.5 Internet of Medical Things (IoMT)

The Internet of Medical Things (IoMT) is a subset of the broader Internet of Things (IoT). It revolves around interconnecting various medical smart devices to create a network that provides many essential healthcare services. IoMT holds tremendous promise in revolutionizing how we approach healthcare, offering a range of advantages spanning remote patient care to enhanced diagnostic capabilities. At its core, IoMT leverages the power of connectivity to create an ecosystem where medical devices, from wearable health monitors to sophisticated diagnostic equipment, can communicate and collaborate seamlessly. This interconnectedness paves the way for numerous advantages that can revolutionize healthcare as we know it. One of the most significant benefits is the provision of remote patient care. IoMT enables healthcare professionals to remotely monitor patients' vital signs, chronic conditions, and overall health status. It enhances the quality of care and reduces the need for patients to make frequent and often time-consuming in-person visits to healthcare facilities. Furthermore, IoMT dramatically improves the speed and efficiency of diagnosis. With real-time data collection and analysis, healthcare providers can swiftly identify and respond to medical issues, leading to faster and more accurate diagnoses. It, in turn, can significantly improve a patient's outcomes and reduce the burden on healthcare systems.

Despite the benefits, IoMT has a significant caveat: the inherent vulnerability of such interconnected systems to cyberattacks. Despite these promising advantages, IoMT is not immune to the perils of the digital age. The interconnected nature of IoMT devices makes them appealing targets for cybercriminals seeking to exploit vulnerabilities for their gain. The potential consequences of inadequate security measures are dire, as they can result in severe patient privacy breaches. Among the concerns, leaking sensitive patient information, including payment details, is a particularly alarming possibility. Unauthorized access to this information can lead to financial losses, identity theft, and profound breaches of trust within the healthcare system. Given these vulnerabilities and risks, it is paramount to prioritize security in IoMT implementations. Robust cybersecurity measures must be integrated into every aspect of IoMT devices and networks. It includes encryption of

data transmission, authentication protocols to ensure only authorized users can access the system, and constant monitoring for suspicious activities. Additionally, healthcare organizations should stay vigilant by keeping software and firmware up to date to patch known vulnerabilities. Therefore, an effective implementation of IoMT necessitates robust emphasis on security measures to safeguard patient information and privacy, preventing potential breaches such as the leakage of sensitive payment details.

A multidisciplinary approach is required to ensure the successful adoption of IoMT while safeguarding a patient's privacy. Collaboration between technology experts, healthcare professionals, and policymakers is essential. Moreover, regulatory frameworks must evolve to address the unique challenges IoMT poses, setting clear security and privacy standards. IoMT holds immense potential to revolutionize healthcare by enabling remote patient care and enhancing diagnostic capabilities. However, this promising technology has its challenges. The interconnected nature of IoMT devices makes them attractive targets for cyberattacks, threatening patient privacy and security. To fully realize the benefits of IoMT while mitigating these risks, a strong emphasis on security measures and collaboration among stakeholders is essential. By incorporating IoMT smart devices into the healthcare infrastructure in a secure manner, we can harness the power of connectivity to improve patient care without compromising privacy and data security.

1.6 Inter-Planetary File System (IPFS)

Inter-Planetary File System (IPFS) [60] is a protocol and peer-to-peer file sharing network to store files in a decentralized database. It uses content-based hashing to identify a file in its storage uniquely. Data is divided into small chunks of 256 KB (known as IPFS objects) and stored on its decentralized system. It also provides versioning of the files. Whenever a file is modified, a separate copy of the modified IPFS objects is created while retaining the original ones. It means that IPFS provides immutability to the files stored on it.

1.7 Motivation and objective of the work

PHRs have always been high-value targets for cyber-attacks; therefore, securing them proves to be a significant challenge. The ability to share health records with selected

parties, such as doctors and insurance companies, further augments the risk of health data leakage. It has led to the requirement of schemes that can not only store health records securely but also allow the secure sharing of health data. Such a scheme requires a guarantee of *confidentiality* of the PHRs, *verifiability* of the correctness of the results returned to the end user [1] and *efficiency* in terms of time and space complexity. Moreover, with the increase in usage of the IoMT, it has become crucial to integrate the healthcare infrastructure with IoMT securely.

Over the years, many solutions have been proposed in the literature for health record storage and sharing [54, 61, 62, 63, 64, 65]. However, some schemes leak too much information during their operations, other schemes do not guarantee the verifiability of the search results, most do not provide any proof for forward security, and most are inefficient because of the use of bilinear pairings, which have high computational costs. Additionally, most of the schemes make use of centralized storage for PHRs, which makes them unreliable in case of an attack on the centralized storage. We identified that most of the existing schemes for PHR sharing use asymmetric SE, and there are not enough works that have explored the usage of symmetric SE for designing a robust PHR sharing scheme with keyword search. Therefore, in this thesis we aim to propose a novel scheme for PHR sharing, by using SSE, blockchain technology and IPFS, that is verifiable, forward secure, efficient, and decentralized.

1.8 Summary of contributions

Our main contribution through this work is to present an efficient PHR-sharing solution by combining SSE, blockchain technology, and IPFS. The existing schemes that use blockchain or IPFS are either inefficient or lack forward security. To the best of our knowledge, no existing schemes have proposed an approach for health data storage and sharing that is verifiable, forward secure, efficient, and decentralized by combining these three. The following are the main contributions made in this thesis work.

- We develop a PHR sharing scheme that maintains the confidentiality of PHRs by proving that it is semantically secure against an adaptive adversary. We prove it to be forward secure and verifiable. Confidentiality is guaranteed by using SSE for

[1] It differs from data integrity, which only ensures that the data is not tampered with after being sent and before being received.

PHRs and other private metadata. By using the "Elliptic Curve Integrated Encryption Scheme (ECIES)" [66] we ensure that communication channels are secure. Verifiability of the results returned to the user is guaranteed by blockchain, and the use of symmetric key encryption ensures that the proposed scheme is efficient. Finally, forward security is ensured by the design of our scheme. Subsequently, we provide rigorous proofs to support our claims as well.

- For the storage of PHRs, we use a decentralized content-addressable file system, IPFS. The hash of its contents uniquely identifies any file stored in IPFS. Our designed scheme relies on this content-hash identity (id) for operation. IPFS has many advantages over centralized storage, such as greater fault tolerance, resistance to Denial of Service (DoS) attacks, and better scalability.

- We then perform extensive experiments using different consensus algorithms, namely "Practical Byzantine Fault Tolerance (PBFT)", Raft, and "Proof of Elapsed Time(PoET)" to demonstrate that our scheme is practical.

- Finally, through the real test-bed experiments, we show that the proposed scheme can be applied in the real-life IoMT-based healthcare applications.

Chapter 2

Mathematical Background

In this chapter, we discuss the following basic mathematical preliminaries that are essential to describe and analyze the security protocols and also for the discussion of the proposed scheme.

2.1 Preliminaries

2.1.1 Pseudo random generator

Suppose f is a function from $\{0,1\}^n \rightarrow \{0,1\}^{l(n)}$, that is, $f : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$, where it takes an n-bit string $x \in \{0,1\}^n$ as an input and produces an $l(n)$-bit string, say $y = f(x)$ $\in \{0,1\}^{l(n)}$ as an output. Now, f is called a pseudo random generator (PRG), if the following properties are valid [67]:

- *Expansion:* For every n, $l(n) > n$. In other words, the length of the output must be greater than that of input.
- *Indistinguishability:* For any "probabilistic polynomial time distinguisher, $\mathcal{D}$", there is a negligible function such that

$$|Pr[\mathcal{D}(G(s)) = 1] - Pr[\mathcal{D}(r) = 1]| \leq negl(n).$$

  Here, the seed $s \in \{0,1\}^n$ is chosen uniformly randomly, $r \in \{0,1\}^{l(n)}$ is chosen uniformly randomly and $Pr[X]$ represents the probability of a random event X. Moreover, the probabilities are considered over the "random coins used by $\mathcal{D}$ and

the choice of r and s". In addition, the function $l(\cdot)$ is known as the "expansion factor of G".

A PRG is then considered as a *deterministic* algorithm that takes a "short truly random seed" and then stretches it into a "long string that is pseudo random".

2.1.2 Pseudo random function

Suppose that $Func_n$ is the set of all functions from $\{0,1\}^n \rightarrow \{0,1\}^n$ and F: $\{0,1\}^* \rightarrow \{0,1\}^*$ is a keyed function that is also length preserving. F is called a pseudo random function (PRF), if there exists a negligible function $negl$, for all probabilistic polynomial time distinguishers $\mathcal{D}$, such that the following property holds [67]:

$$|Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1]| \leq negl(n)$$

Here, the key k is chosen uniformly at random from the set $\{0,1\}^n$ and f is chosen uniformly at random from the set $Func_n$.

In cryptography, the PRFs become a very important building block for a number of various cryptographic constructions. For instance, there are efficient primitives, known as *block ciphers*, that are assumed to act as PRFs.

2.1.3 Bilinear maps

Let G_1 and G_2 represent an "additive cyclic group" and a "multiplicative cyclic group" with a prime order p. Assume that g_1 is a generator of G_1. A mapping $e : G_1 \times G_1 \rightarrow G_2$ is called a "bilinear map", if it fulfils the following properties [68, 69]:

- *Bi-linearity:* $\forall u, v \in G_1$ and $\forall a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$ holds, where $Z_p = \{0, 1, 2, \cdots, p-1\}$.
- *Non-degeneracy:* $e(g_1, g_1) \neq 1$, where the identity in G_2 is 1.
- *Computability:* $\forall u, v \in G_1$, $e(u, v)$ is efficiently computable.

The map e is called as an admissible bilinear map if $e(g_1, g_1)$ generates G_2 as well as e is also efficiently computable.

We have the following computational problems in the bilinear pairing setting.

Definition 2.1 (Bilinear Diffie-Hellman Problem (BDHP)). *If G is a "multiplicative cyclic group that is generated by g of prime order p", and given a tuple (a, g^a, g^b, g^c) for $a, b, c \in Z_p$, compute $e(g,g)^{abc}$.*

Definition 2.2 (Weak Diffie-Hellman Problem (WDHP)). *Assume that G is a "multiplicative cyclic group generated by g of prime order p". If (g, h, g^a) for $a \in Z_p$ are given, compute h^a. The problem is called a weak Diffie-Hellman problem due to the fact that $e(g^a, h) = e(g,h)^a = e(g, h^a)$.*

2.1.4 Collision resistant hash functions

It is a "cryptographic function" that takes an input of a variable length and produces an output of a fixed length, called a hash value or message digest. Mathematically, it can be expressed as a mapping as h: $A \rightarrow B$, where the set $A = \{0,1\}^*$ consists of all the strings of variable length and the set $B = \{0,1\}^{l_h}$ consists of the strings of fixed length l_h [70].

The formal definition of a one way hash function is as follows [70].

Definition 2.3 (Collision-resistant one-way hash function). *A collision resistant one-way hash function h: $\{0,1\}^* \rightarrow \{0,1\}^{l_h}$ is a deterministic function which takes an arbitrary length binary string $\alpha \in \{0,1\}^*$ as input and returns a fixed-length binary string of l_h bits as $\beta = h(\alpha) \in \{0,1\}^{l_h}$, called the output. The advantage $Adv_{\mathcal{A}}^{HASH}(t_p)$ of an adversary $\mathcal{A}$ in finding the hash collision in polynomial time t_p is given by*

$$Adv_{\mathcal{A}}^{HASH}(t_p) = Pr[(\alpha, \alpha') \leftarrow \mathcal{A} : \alpha \neq \alpha' \, and \, h(\alpha) = h(\alpha')]$$

where $Pr[X]$ denotes probability of random event X, and $(\alpha, \alpha') \leftarrow \mathcal{A}$ represents that the pair (α, α') is randomly picked by $\mathcal{A}$. By an "(ψ, t_p)-adversary $\mathcal{A}$ attacking the collision resistance of $h(\cdot)$", we mean that the "runtime of $\mathcal{A}$ is bounded by t_p and that $Adv_{\mathcal{A}}^{HASH}(t_p) \leq \psi$."

A hash function has the following properties [70]:

- One way hash functions are easy to compute, that is, given an input $\alpha \in A$ it is easy to compute $h(\alpha) \in B$, but computing the inverse of the function is computationally

infeasible. Thus, given the hash value $h(\alpha)$ of any input α, the computation of the original input α is computationally infeasible. This property is known as *pre-image resistant*.

- For any given $\alpha \in A$, there exists no feasible way in polynomial time to find out another input $\alpha' \in A$ such that $h(\alpha) = h(\alpha')$. This property is called *weak collision resistant property*.

- It is computationally infeasible to find any two inputs $\alpha, \alpha' \in A$, such that $\alpha \neq \alpha'$ with $h(\alpha) = h(\alpha')$. This property is known as *strong collision resistant property*.

Some examples of secure collision-resistant hash functions include "Secure Hash Algorithm (SHA-1) and its variants SHA-224, SHA-256, SHA-384, and SHA-512 [71]."

2.1.5 Bloom filters

A Bloom filter [72] is a data structure that is used to test set membership of an element. It can give false-positive results with some probability but never False-negatives. It consists of a set of n elements $S = \{s_1, \ldots, s_n\}$ represented by array of m bits. All the bits of array are initially 0. A set of r independent hash functions $h_1, \ldots, h_r$ are used by the filter where $h_i : \{0,1\}^* \rightarrow [1, m] \forall i \in [1, r]$. The bits at position $h_1(s), \ldots, h_r(s)$ of the array are set to 1 for each element $s \in S$. We can set a location to 1 multiple times, however, only the first one is significant. Further, to determine if set S contains an element a, we check whether all the bits at positions $h_1(a), \ldots, h_r(a)$ are 1 or not. If they are all 1s, then $a \in S$. As a result, with some probability false positive may occur because some other element other than a may also set the same location.

2.1.6 Standard blockchain model

To formally analyze security schemes that use blockchain, Kosba *et al.* [73] proposed the standard blockchain model. It treats blockchain as a trusted entity that is guarantees correctness and availability but cannot be trusted for privacy. This is because all the transactions can be read by every participating node thus privacy cannot be guaranteed by the blockchain unless the transactions are encrypted in some form. However, without the consensus of majority of nodes a transaction cannot be modified, therefore, cannot be modified by them.

2.1.7 Dictionary

A dictionary is an abstract data type that is used to store key-value pairs such that every key occurs only once. It is also known as *map*, *associative array* or *symbol table*. A dictionary must support the following basic operations:

- *Insertion:* It requires a key-value pair as its input and stores it in the dictionary. The key-value pair is also called an element of of the dictionary. If an element with the same key exists, then its value is overridden by the new input value.
- *Deletion:* It requires the element's key to be deleted as the input and removes the corresponding key-value pair from the dictionary.
- *Lookup:* It requires a key as its input and returns the corresponding value. An error or default value is returned if the key is not present in the dictionary.

2.1.8 Inter-Planetary File System (IPFS)

It is a content-based distributed file system [60] where the data is identified through the hash of the contents rather than giving them an independent identifier. IPFS is also a decentralized protocol designed to change how we store and share data online. Unlike traditional web hosting, where data is stored on centralized servers, IPFS utilizes a distributed network of nodes to store and retrieve content. Each file is given a unique cryptographic hash, making it immutable and resistant to censorship. IPFS enhances data security and improves content delivery, as files can be cached locally, reducing latency. This technology can potentially create a more resilient and user-centric internet, where information is accessible, permanent, and free from the vulnerabilities of traditional server-based systems.

2.1.9 Hyperledger sawtooth

It is a blockchain framework maintained by Hyperledger foundation and provides a flexible solution for developing decentralized applications [74]. It allows the programmers to implement application logic in a variety of languages, including Python and JAVA unlike other platforms such as Ethereum, which restricts the programmer to Solidity. Each node in a Sawtooth network runs a process known as *transaction processor*. It needs to be implemented by the application developer for processing the transactions of his

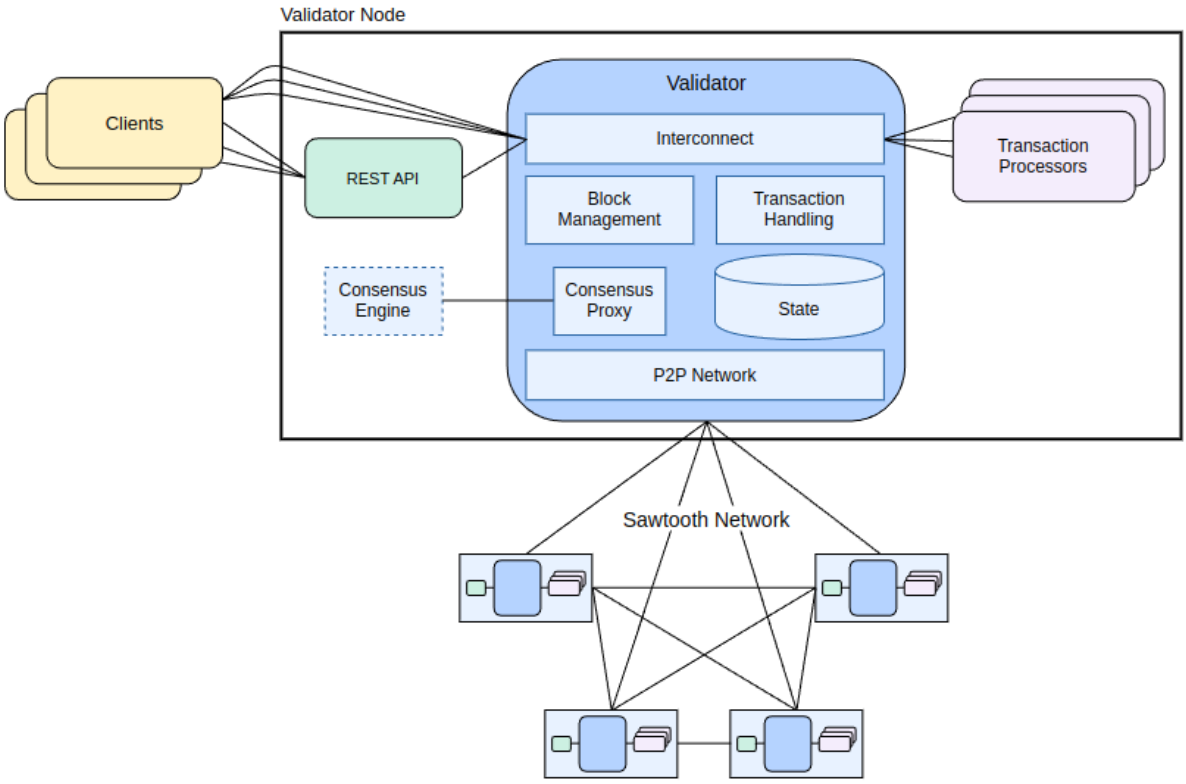

Figure 2.1: Hyperledger sawtooth architecture (Adopted from [2])

application. A sawtooth node can run multiple transaction processors for different families of transactions. Figure 2.1 shows the architecture of the sawtooth with five nodes. The different components of each node are described below.

- *Validator:* Responsible for validating transactions and proposing new blocks to the network.
- *Transaction Processor:* Executes transactions within the node, interacting with the Transaction Execution Engine.
- *Consensus Engine:* Coordinates the chosen consensus algorithm for the network.
- *Transaction Execution Engine (TEE):* Executes smart contracts or Transaction Families, updating the global state.
- *REST API:* Provides an interface for submitting transactions and querying the blockchain state.

- *Network Interface:* Handles communication with other nodes, facilitating transaction and block propagation.
- *Block Publisher:* Creates new blocks by selecting a set of valid transactions.
- *Identity and Access Management:* Manages participant identity and controls access within the network.
- *Configuration Management:* Manages node-specific configuration settings.
- *Event System:* Allows monitoring of blockchain events with subscription capabilities.
- *Consensus API Interface:* Provides an interface for integrating custom consensus algorithms into the node.

2.1.10 Merkle-Radix (MR) tree

A Merkle Radix tree [3], sometimes called a Merkle Patricia tree, is a data structure used in blockchain and distributed ledger technologies to store and verify large data sets efficiently. It is an extension of the more common Merkle tree structure, designed to work effectively with key-value pairs, particularly in cryptocurrencies like Ethereum. The Merkle Radix tree is well-suited for scenarios with similar keys, such as in blockchain or distributed ledger systems, where accounts or contracts may have addresses that share common prefixes. It efficiently stores and verifies the state of these accounts and provides a secure and compact representation of the data. Figure 2.2 shows the structure of merkle radix tree for hyerledger sawtooth.

Given below are its components and its working:

- *Radix tree structure:* At its core, the Merkle Radix Tree is a radix tree, a tree structure used for representing strings, where common prefixes of keys are shared among nodes. It is handy for efficiently storing and retrieving key-value pairs with similar keys (common prefixes).
- *Merkle tree hashing:* Each node in the tree contains a cryptographic hash of its children. It is similar to a regular Merkle Tree, where leaf nodes are data blocks, and each non-leaf node is a hash of its children. In the case of Merkle Radix Trees,

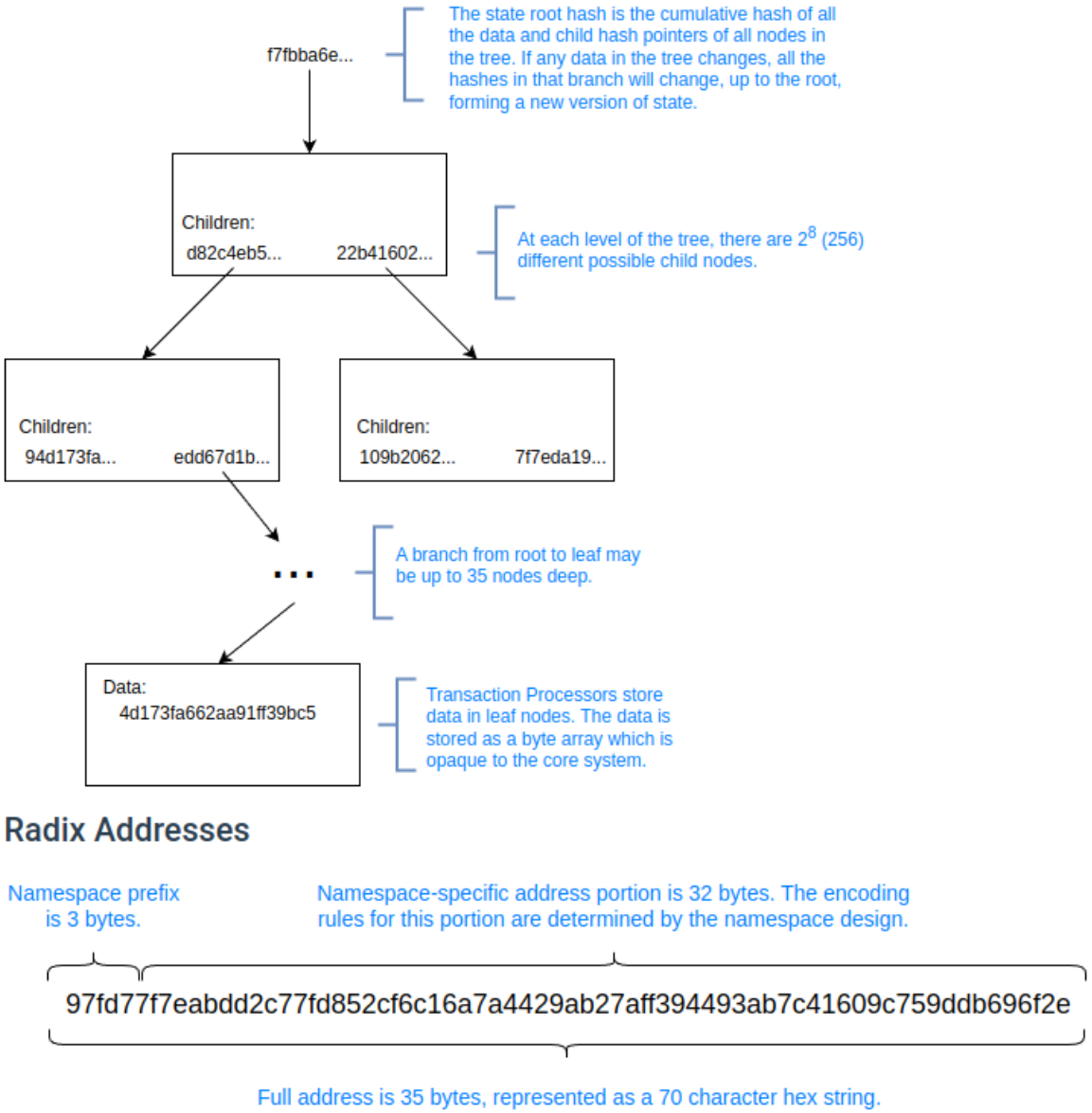

Figure 2.2: Merkle Radix tree used in hyperledger sawtooth (Adopted from [3])

the nodes might represent a series of characters in the key, and the cryptographic hash is calculated over the concatenation of the hashes of child nodes.

- *Key-Value storage:* Besides the hash values, each node in the Merkle Radix Tree stores a partial key and its associated value. The partial key is a segment of the full key. It allows for efficient key-value storage and retrieval and the ability to look up keys with common prefixes without duplicating information.

- *Path compression:* One of the main features of the Merkle Radix Tree is its ability to compress common prefixes, significantly reducing the amount of storage required. When keys share a common prefix, the tree structure allows these prefixes to be stored only once, while differentiating values are stored in separate branches.

- *Cryptographic security:* Using cryptographic hashes at each tree level ensures data integrity and security. Any change in the data at any tree level would result in a different root hash, making it easy to detect tampering.

2.1.11 Elliptic curve integrated encryption scheme (ECIES)

The Elliptic Curve Integrated Encryption Scheme (ECIES) is a hybrid encryption scheme that combines the properties of elliptic curve cryptography (ECC) and symmetric-key cryptography to provide secure and efficient public-key encryption. It was proposed by Victor Shoup [66]. ECIES uses an elliptic curve key agreement protocol, such as ECDH, and a symmetric encryption scheme. The key agreement protocol derives the shared secret for the sender, which is then used to encrypt the data to be transferred. The encrypted data is sent to the receiver along with the information by which the receiver can derive the shared secret. The receiver can then use this information to recover the original plaintext. ECIES is commonly used to secure data in transit, such as communications and encryption for technologies like blockchain and digital currencies. ECIES offers several advantages, including robust security, efficient encryption and decryption, and the ability to secure data so that only the intended recipient can access it. It's widely used in various secure communication protocols, including secure messaging apps and the encryption of cryptocurrency transactions in blockchain systems. ECIES is just one example of a hybrid encryption scheme, and the exact implementation may vary depending on the specific use case and the cryptographic libraries or protocols being used. Given below is a brief overview of the components and working of ECIES:

- *Elliptic Curve Cryptography (ECC):* ECC is a type of public-key cryptography that uses the mathematics of elliptic curves to provide strong security with relatively short key lengths. ECIES leverages ECC for the key exchange phase of encryption. It allows the two parties (sender and receiver) to establish a shared secret key without revealing sensitive information to each other.

- *Symmetric-key cryptography:* ECIES uses symmetric-key cryptography for the actual encryption and decryption of the data. A shared secret key, derived from the ECC key exchange, is used to encrypt and decrypt the message.

- *Key derivation function (KDF):* A key derivation function derives the shared secret key from the ECC key exchange. It ensures that the key used for symmetric encryption is unpredictable and secure.

- *Encryption:* The sender uses the derived shared secret key to encrypt the plaintext message with a symmetric encryption algorithm (e.g., Advanced Encryption Standard (AES) algorithm [75]). This symmetric encryption is often faster and more efficient for encrypting the message.

- *Message authentication code (MAC):* To ensure the integrity and authenticity of the message, a MAC is often generated from the plaintext message and attached to it. The MAC is used to detect any tampering with the message during transit.

- *Recipient public key:* The sender typically encrypts the symmetric key used for encryption with the recipient's public key to ensure that only the recipient can decrypt the message.

- *Decryption:* The recipient, who possesses the corresponding private key for their public key, can use it to decrypt the symmetric key. With the symmetric key in hand, they can decrypt the message and verify the MAC.

2.1.12 AES-OCB3

The "Advanced Encryption Standard Offset Codebook v3 (AES-OCB3)" [75] is a symmetric key encryption algorithm that is the result of using the "Advanced Encryption Standard (AES)" in Offset Codebook v3 (OCB3) mode. OCB mode integrates MAC with the operations of AES in block-cipher mode. It eliminates the need for explicitly

using an authentication mechanism. by providing encryption and data authentication in a single pass. AES-OCB3 is known for its efficiency, as it combines encryption and authentication into one step, reducing computational overhead compared to separate encryption and authentication modes. It was designed to offer strong security while minimizing complexity and computational requirements.

AES-OCB3 is used in various applications requiring authenticated encryption, such as in network security protocols, secure messaging, data storage, and more. However, as with any cryptographic system, it is crucial to implement it correctly and securely, including properly handling keys and nonces, to ensure its effectiveness in practice. Following are the key features of AES-OCB3:

- *Authenticated encryption:* AES-OCB3 encrypts data while simultaneously generating an authentication tag that ensures the integrity of the ciphertext. This makes it suitable for securing both confidentiality and data integrity.
- *Low overhead:* Unlike other encryption modes, AES-OCB3 requires minimal additional data for authentication. This results in less overhead and a smaller ciphertext size.
- *Parallel processing:* OCB3 is designed to be parallelizable, making it suitable for hardware acceleration and efficient on modern computer systems.
- *Security:* AES-OCB3 is considered secure and has undergone extensive cryptanalysis. It is designed to protect against various cryptographic attacks, including plaintext attacks and ciphertext attacks.
- *Speed:* It is known for its high speed and efficiency, making it suitable for applications where encryption and authentication must be performed quickly.
- *Patent-free:* Unlike some encryption modes, AES-OCB3 is not encumbered by patents, making it freely available for use in various software and systems.

2.2 Summary

This chapter discussed various mathematical preliminaries required for understanding PHR sharing and storage using searchable encryption, blockchain, and IPFS. We discussed low-level primitives, such as pseudo-random generators and pseudo-random functions and

relatively high-level primitives such as Advanced Encryption Standard (AES) and Elliptic Curve Integrated Encryption Scheme (ECIES). Such primitives are the base of any security scheme; thus, a good understanding of them is necessary. Additionally, we discussed the data structures required, namely, dictionary, Bloom filter, and Merkle radix tree. We also discussed some of the frameworks we have used in our work, namely, Hyperledger sawtooth and IPFS. These preliminaries are building blocks for securing and efficiently managing sensitive healthcare data. In conclusion, this chapter equipped us with the foundational knowledge and tools to delve deeper into the intricacies of PHR sharing and storage using cutting-edge technologies, like searchable encryption, blockchain, and IPFS.

Chapter 3

Literature Survey

In this chapter, we will explore various existing works that have been done over SSE and blockchain. We will then provide a detailed overview of how these two technologies have been combined over the years to provide solutions for the storage and sharing of health data, and list their pros and cons. To make this thesis easier to read, we have provided a list of abbreviations used, in Table 3.1.

3.1 Existing surveys

Over the years, several survey works over searchable encryption (SE) have been done. We present the findings of these survey works in this section and also compare them with our survey.

- **Bosch *et al.* [76] (2014):** This survey gives an overview of the works in searchable encryption done till 2014. They have categorized the different SE schemes under four categories - 1) "*Single Writer/Single Reader*", 2) "*Single Writer/Multiple Reader*", 3) "*Multi Writer/Single Reader*", and 4) "*Multi Writer/Multi Reader*". Here, the writer refers to data owner who generates the data to be secured and the reader refers to data users. This is a detailed survey of SE schemes; however, it does not include any SE schemes that make use of blockchain. The main reason for this is that till 2014 there were hardly any works in the direction of SE and blockchain.

- **Wang *et al.* [77] (2016):** This survey is mainly focussed on two of the standard SE techniques - "*Searchable Symmetric Encryption (SSE)*" and "*Public Key En-*

Table 3.1: Abbreviations

Abbreviation	**Full Form**
ABKS	Attribute Based Encryption with Keyword Search
API	Application Programming Interface
BN	Blockchain Node
CHD	Current Head Dictionary
D	Doctor
DO	Data Owner
CS	Cloud Server
DU	Data User
ECDH	Elliptic Curve Diffie-Hellman
ECIES	Elliptic Curve Integrated Encryption Scheme
H	Hospital
HS	Hospital Server
IPFS	Inter Planetary File System
MR	Merkle Radix
P	Patient
PEKS	Public Key Encryption with Keyword Search
PHR	Personal Health Record
PRES	Proxy Re-encryption with Keyword search
REST	Representational State Transfer
SE	Searchable Encryption
SSE	Searchable Symmetric Encryption

cryption with Keyword Search (PEKS)". The authors have compared the schemes based on index size, search time, security and dynamism (for SSE), and security assumptions (for PEKS).

- **Poh *et al.*** [78] **(2017):** This is a detailed survey that gives a good insight into the

workings of SE schemes. The authors have categorized the existing schemes based on principals involved, operations, security models, entity setups, query functionalities, cryptographic primitives, scheme structure, performance and characteristics. It is targeted towards readers having intermediate background in security. It indeed covers most of the literature of SE; however, it does not include schemes that make use of blockchain.

- **Zhang *et al.* [79] (2018):** In this survey, the authors explore the application of SE in healthcare industry. They have presented four scenarios in which SE can be used to secure health records, namely, owner as reader/writer, Single reader/Single writer, One writer/many readers, authorization delegation. Reader and Writer refer to health data producer and health data consumer. They also provide an overview of the standard SE techniques, namely, "*Searchable Symmetric Encryption (SSE)*", "*Attribute Based Encryption with Keyword Search (ABKS)*", "*Public Key Encryption with Keyword Search(PEKS)*" and "*Proxy Re-encryption with Keyword Search(PRES)*". This work, however, lacks a discussion and comparison of existing SE schemes for healthcare and also does not describe schemes that use blockchain.

- **Pham *et al.* [80] (2019):** In this work, the authors have provided a general overview of the system architecture involved in a SE scheme. They have categorized the schemes, based on the type of search, into - 1) "*Keyword Search*", 2) "*Regular Expression Search*", and 3) "*Semantic Search*". On the basis of security levels they have categorized the schemes into - 1) *Somewhat secure*, 2) *Semi secure*, 3) *Secure* and 4) *Fully Secure*. This work also explores the application areas of SE such as healthcare, law enforcement and text editors. However, this work too does not include SE schemes that use blockchain.

- **Andola *et al.* [81] (2022):** This work is geared towards survey of application of SE in storage in cloud. On the basis of the search type the authors have classified SE on the cloud into "sequential search, index-based search, rank based search, fuzzy keyword search, conjunctive keyword search and access-control-based search". Further, they have also discussed cryptographic primitives used in these categories of SE. However, on downside, no SE schemes using blockchain have been covered in this survey. We note that this paper requires reader to have basic security background.

Year	Survey	Scope	Requirement
2014	Bosch *et al.* [76]	SE	Basic Security Background
2016	Wang *et al.* [77]	SSE and PEKS	Basic Security Background
2017	Poh *et al.* [78]	SE	Intermediate Security Background
2018	Zhang *et al.* [79]	SE for Healthcare	Intermediate Security Background
2019	Pham *et al.* [80]	SE System Architecture	Basic Security Background
2022	Andola *et al.* [81]	SE with cloud storage	Basic Security Background

Table 3.2: Existing surveys

In summary, in Table 3.2, we have provided the above discussed survey works with their scopes and requirements.

3.2 Related works

3.2.1 Searchable encryption

Song *et al.* [59] in 2000 took the first step toward constructing an efficient searchable encryption scheme. Earlier works [82] made use of Oblivious RAMs (ORAMs), and gave a complete and general solution to the problem of searchable encryption. However, ORAMs are quite expensive in terms of time and space. Therefore, researchers started looking for alternative solutions that would be efficient. The scheme proposed by *Song et al.* [59] encrypts each word in the file using a deterministic algorithm. Later during search operation, a linear scan is performed over the encrypted file to search for keywords. Although the work in [59] is more efficient than ORAM, it was linear in the total number of keywords in document collection, and hence, it is unsuitable for practical applications.

To overcome these limitations, Goh *et al.* [83] in 2003 proposed an alternative solution using an index based approach. They constructed a secure inverted index and demonstrated its application in SE through a scheme with constant search time. In their proposed scheme the data owner generates a secure index over the files that are to be encrypted. This secure index is uploaded to the cloud server along with the data files which are encrypted using some standard symmetric encryption algorithm such as AES.

They also introduced a security model known as "semantic security against adaptive chosen keyword attack (IND-CKA) and its stronger variant IND2-CKA." In IND-CKA, A challenger C gives an adversary A two documents V_0 and V_1 such that their lengths are equal, along with an encrypted index. The adversary A is then challenged to determine which document is encoded in the index. Hardness of deducing whether the index is of V_0 or V_1 implies the hardness of using the index to find a word such that V_0 and V_1 do not have it in common. In the same work they also proposed Z-IDX, which is an efficient IND-CKA secure index construction, using "PRFs" and "Bloom Filters". To make search time linear in terms of number of documents, a bloom filter is created for each document in the proposed scheme. Their scheme, unfortunately, had a weak security model which was subsequently improved upon by Curtmola *et al.* [84] in 2006.

Searchable encryption was extended to public key setting in 2004 by Boneh *et al.*[85] who took a mail server as the reference for their scheme and proposed the first public key encryption with keyword search (PEKS) scheme. PEKS uses Identity based encryption (IBE) [86] with the keyword as the identity. The sender encrypts his/her message using any standard public key system such as RSA or EC and then appends the PEKS of each keyword $w_1, w_2, w_3, \ldots, w_m$ with it. To obtain the PEKS, a publicly known string is encrypted using public key associated with keyword as identity. The ciphertext obtained is of the form shown below:

$$E_{K_{pub}} \| C_1 = PEKS(K_{pub}, w1) \| \ldots \| C_m = PEKS(K_{pub}, w_m)$$

To search for a keyword the receiver derives a secret key from the keyword it wants to search using the master key. This secret key is the trapdoor that is sent to the server, which then attempts to decrypt the IBE cipher-texts. If the decryption results in a publicly known string, then it is successful and it can be concluded that the attacked encrypted message contains the keyword.

Previous works in SE lacked a robust security model so in 2005 Chang *et al.* [87] proposed a real-ideal simulation model which is now widely used in subsequent works over searchable encryption.

Later, in 2006, Curtmola *et al.* [84] proved that IND-CKA and IND2-CKA security models were not strong enough and introduced two new security models for SSE, namely, *non-adaptive model* and *adaptive model*, which become the standards for securing SE schemes, and also provided two schemes: SSE-1 and SSE-2 against the proposed security models respectively. However, their scheme was static, and therefore it had limited practical applications.

Next major breakthrough in SE was in 2012, when Kamara *et al.* [88] extended the work of Curtmola *et al.*and introduced "*dynamic searchable encryption (DSE)*" scheme which allows addition and deletion of new files efficiently.

Even though a lot of research on SE had been done by that time, Islam *et al.* [89] pointed out that there had been no proper study of attacks on SE schemes. Their work showed that various attacks were possible due access pattern disclosure in SE schemes and emphasised on the need of *forward security* on SE schemes. Forward security refers to the notion that an adversary should not learn anything new when a new file is added. There was a growing need for forward security after Islam *et al.*'s work. Based on this Stefanov *et al.* [90] presented the first forward secure SE scheme in 2013. They attempted to find a balance between security and efficiency through their scheme. However, due heavy dependency on client side computation, its practical applications were limited. Advancing in the direction of efficient searchable encryption Kamara *et al.* [91] proposed a Parallel and "dynamic searchable encryption scheme". In this they have used a *Red-Black tree* to create a *keyword Red Black (KRB)* tree data structure.

Nevertheless, scalability was still an issue as most of the schemes supported only single keyword search. Working in this direction, Cash *et al.* [92] introduced a scheme that supported conjunctive keyword search along with the Boolean queries. Another issue that needed to be addressed in SE was that none of the works had ever attempted to formalize the data leakage occurring during the operations. Cash *et al.* [93] in 2015 provided formalization of leakages in SE schemes along with possible attacks due to these leakages, for the first time. Prior to that, no other work had focused on the study of attacks on SE schemes. It formalized the leakages and classified the existing schemes based on four levels of leakage.

Later, it was identified that in order to save the computational costs, a server can return partial or wrong results to the data user. To address this issue, the requirement of verifiability was added to SE schemes. Therefore, in 2016 the notion of *verifiability* was introduced by Bost *et al.* [94] by improving Stefanov *et al.*'s scheme. Verifiability refers to whether the results returned to the end user by the cloud server are correct or not. For example, during lack of verifiability a server can return partial results to the end user to save computation costs.

With time, the need for more features was identified in SE schemes – one of which was backward privacy. It ensures that after deletion of data, future queries cannot be performed on it. The first scheme to support *backward privacy* was introduced by Bost

et al. [95] in 2017. Backward privacy refers to the leakage that can occur after deletion of a file from the server that was previously queried. In the same year, Kim *et al.* [96] introduced a dynamic SE scheme with a primary focus on data deletion, which previous schemes mainly had overlooked. In the following year, an efficient, forward secure and parallelizable DSE scheme was proposed by *et al.* [97] which has performance on par with previous non-forward secure DSE schemes. However, a security flaw in backward privacy for a special case was discovered in Etemad's scheme in 2022 by Watanabe *et al.* [98] which was also patched by them.

Based on these works we observed that there is a trade-off between security and efficiency in searchable encryption schemes. These trade-offs have been presented in the work by Ashrov *et al.* [99].

Many of the subsequent works [96, 100, 101, 102, 103, 104] have focused on providing schemes that guarantee forward and backward secrecy but also efficiency at the same time. Additionally, there are other schemes, such as the scheme proposed in [105] which have focused on providing additional features such as data de-duplication and conjunctive queries while maintaining required properties of forward and backward security.

3.2.2 Blockchain-based SE schemes

As discussed earlier, for practical applications of searchable encryption, there is a need to verify whether the server returns correct results or not. A malicious server could return the wrong results in order to save costs and still receive full payment from the user. Many of the works since 2018 have incorporated blockchain into SE schemes to provide verifiability. Hu *et al.* [8] proposed the first work that made use of blockchain in searchable encryption in 2018. They eliminated the use of a central server for data storage and instead used Blockchain as its replacement. Their work aimed to introduce a searchable symmetric encryption scheme in which the results returned by a potentially malicious server are verifiable by the data user. By utilizing smart contracts they provided a solution in which the results returned to the user were always correct and all the parties received the correct payment. This scheme, however, has a significant shortcoming in that Blockchain is not designed to handle a large amount of data and using it to store large files results in impractical latency.

In the same year, the work of Cai *et al.* [5] sought to prevent dishonesty both at client and server side regarding payment for search results by recording the logs of transactions

in a public Blockchain. This scheme, however, requires explicit verification of the results by the client from the blockchain.

Since then several schemes [6, 7] have been proposed that did not eliminate the central server but instead used Ethereum smart contracts to ensure fairness between client and server. The smart contract is designed in such as way that until both parties agree, the payment will not be released.

To enable forward security in blockchain-based searchable schemes, in 2020, Guo *et al.*[106] suggested a verifiable and forward secure encrypted search which also leveraged the blockchain technology. In the same year Du *et al.*[107] also published a similar work that provided forward as well as backward secrecy along with verified results.

3.2.3 PHR sharing using blockchain

After the success of Bitcoin which was proposed in 2008 [11], the researchers began to find the application of blockchain in other areas, one of which was healthcare. An early work for application of blockchain in healthcare was proposed by Azaria *et al.* [108] in 2016. It focused on taking advantage of transparency and immutability of blockchain by storing references of medical data onto blockchain and leverage its to ensure data integrity, verifiability, and data sharing. Another work proposed by Yue *et al.* [109] focused on an app called "Healthcare Data Gateway (HDG)" which uses blockchain to let users exercise full control over their medical records.

Subsequently, one of the early works that specifically targeted secure sharing of health records by using immutability and autonomy of blockchain was proposed by Xia *et al.* [110] in 2017. They attempted to address the access control problems associated with health records using the immutability and autonomy provided by blockchain.

Next, Fan *et al.* [111] improved upon the scheme proposed by Xia *et al.* [110] by providing efficient access and retrieval mechanisms and addressed the problems related to privacy originated by sharing health data with third parties.

3.2.4 PHR sharing using SE and blockchain

Zhang *et al.* [54] in 2018 proposed that used blockchain as well as searchable encryption for sharing of health records. Their scheme, called "Blockchain-based Secure and Privacy-preserving PHI sharing (BSPP)", that makes use of both private blockchain and consortium blockchain along with public encryption with keyword search (PEKS) to build

blockchain-based secure and privacy-preserving PHI sharing (BSPP) scheme. However, it is inefficient due to heavy dependency on bilinear pairing.

Next in 2021, Wang *et al.* [61] made use of a decentralized storage called *Inter-planetary File System (IPFS)* to store the PHRs. However, these schemes rely on bi-linear pairing which increases its computation time. In the past few years, there have been works [62, 63] that make use of proxy re-encryption based SE for health record sharing. Yet again, these schemes rely on bilinear pairings thereby requiring higher computation power. On the basis of SSE there have been lesser number of for health record sharing such as that of Chen *et al.* [64] and the one proposed by Tang *et al.* [65].

In addition, some of the schemes [112, 113, 114, 115] have used attribute-based encryption to have better access control over sharing of PHR. However, all these schemes rely on bilinear maps for security, making them inefficient. Additionally, there has not been any focus on forward security in these schemes, and many are not dynamic. Unlike the schemes discussed so far, recently, in 2021, Tang *et al.* [65] proposed a PHR sharing scheme based on searchable symmetric encryption (SSE) and blockchain. However, it requires the concerned hospitals to store the data on a local server, which is not a realistic assumption. A recent scheme proposed by Nie *et al.* [116] uses IPFS to store the encrypted cipher-texts and blockchain for meta-data. However, it is not dynamic in nature, i.e., all the health records are encrypted at a time and there is no mechanism to add the health records later. We have described these scheme in detail later in Section 3.4.

3.2.5 Important observations

We observed that there are many schemes that make use of asymmetric searchable encryption for health data sharing. However, the use of symmetric searchable encryption for the same has not been explored well enough, even though the literature of SSE is quite strong. Our work tries to fill this research gap by providing a symmetric searchable encryption based health data storage and sharing scheme using blockchain technology and IPFS. The proposed scheme is able to provide confidentiality of PHRs and other private metadata, secure communication channels, verifiability using blockchain, forward security, and also efficiency using symmetric key encryption.

3.3 Searchable encryption in detail

3.3.1 System models for SE

Here we discuss the different types of system models employed by researchers for searchable encryption. Broadly, there are two types of systems that have been used in the literature. First one is the traditional SE system model which involves a data owner, cloud server and data user. Second one additionally uses blockchain to provide more features such as verifiability.

In the following, we describe both models: 1) SE without blockchain and 2) SE with blockchain, and the benefits of using SE with blockchain over SE without blockchain.

- *SE without blockchain:* Figure 3.1 shows the general system model for searchable encryption when blockchain is not involved. There are three entities in this model - 1) Data Owner (DO), 2) Data User(DU) and 3) Cloud Server (CS). First, the data owner generates an encrypted index and encrypted files from the plaintext files and sends them to the could server. The encrypted index is used by the cloud server to search for documents containing a given keyword. To search for a keyword, the data user requests the data owner to for a token/trapdoor, which it then sends to the cloud server. The cloud server upon receiving the trapdoor searches for any document containing the keyword hidden in the trapdoor and returns the results to the user.

- *SE with blockchain:* By utilizing the immutability property of blockchain, searchable encryption schemes which have intrinsic verifiability can be constructed. There are multiple ways in which blockchain has been used along with searchable encryption. However, here we have shown two of those ways. Figure 3.2 shows the first method in which the cloud server is entirely replaced with blockchain. The encrypted documents as well as the encrypted index are all stored on the blockchain. Hu *et al.* [8] were the first to use this technique.

  The second method has been show in Figure 3.3. In this method, the data is stored on a storage server which can be cloud or a distributed file system such as IPFS [117]. Blockchain is used to store metadata such as encrypted index. This reduces the overhead of processing large files on the blockchain. The data owner first encrypts the files/documents and uploads them to the storage server. The storage server provides him the unique file identifiers for the uploaded files. The

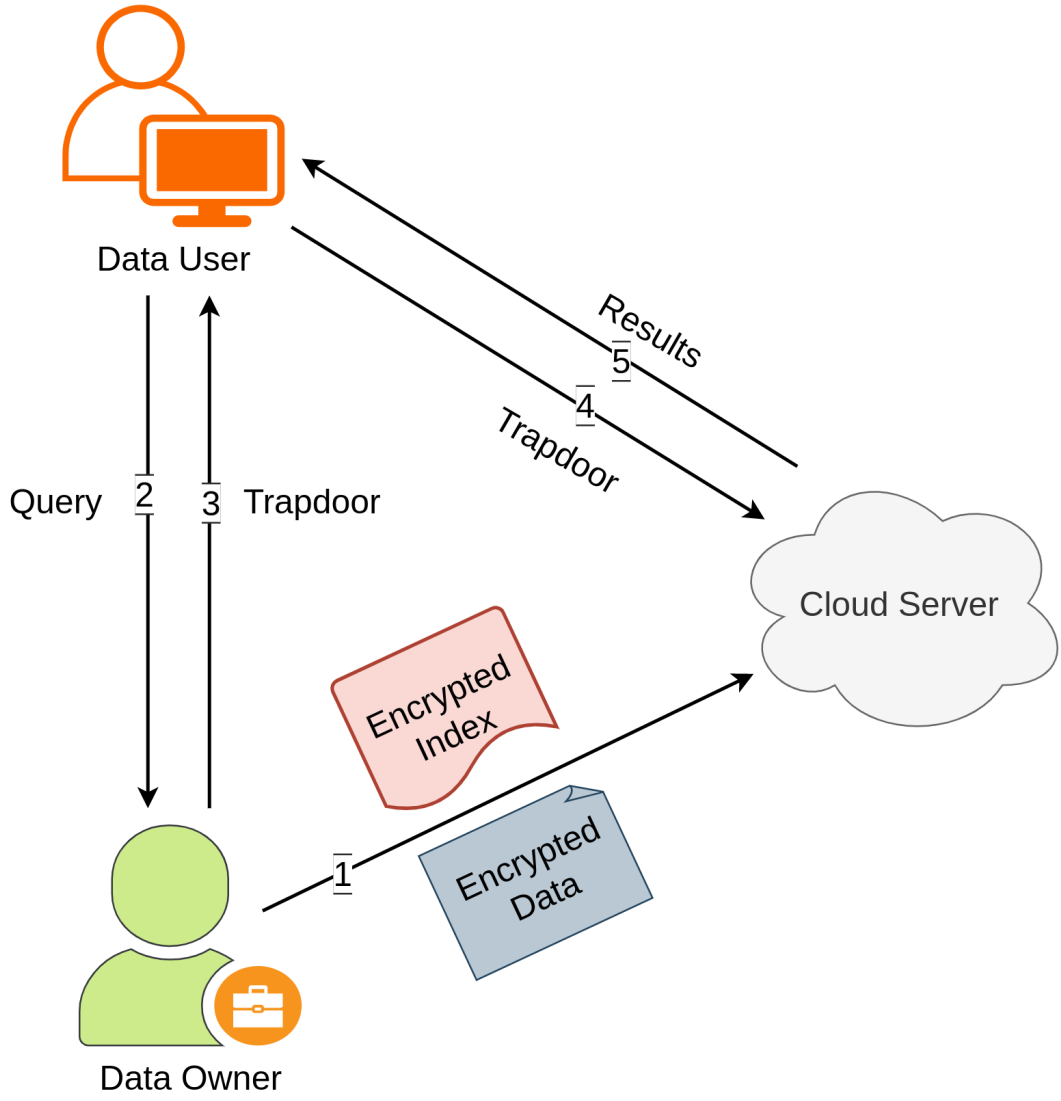

Figure 3.1: Searchable encryption without blockchain

DO then uses these identifiers and the original files to create an encrypted index. This encrypted index is then uploaded to the blockchain.

When a user wants to search for a keyword, he requests the data owner for a trapdoor corresponding to his/her query. The user then sends this trapdoor to the blockchain to obtain the file ids which contains the keyword. Finally, the user uses the received file ids to download the files from the storage server. A typical use of this technique can be seen in the paper by Guo *et al.* [106].

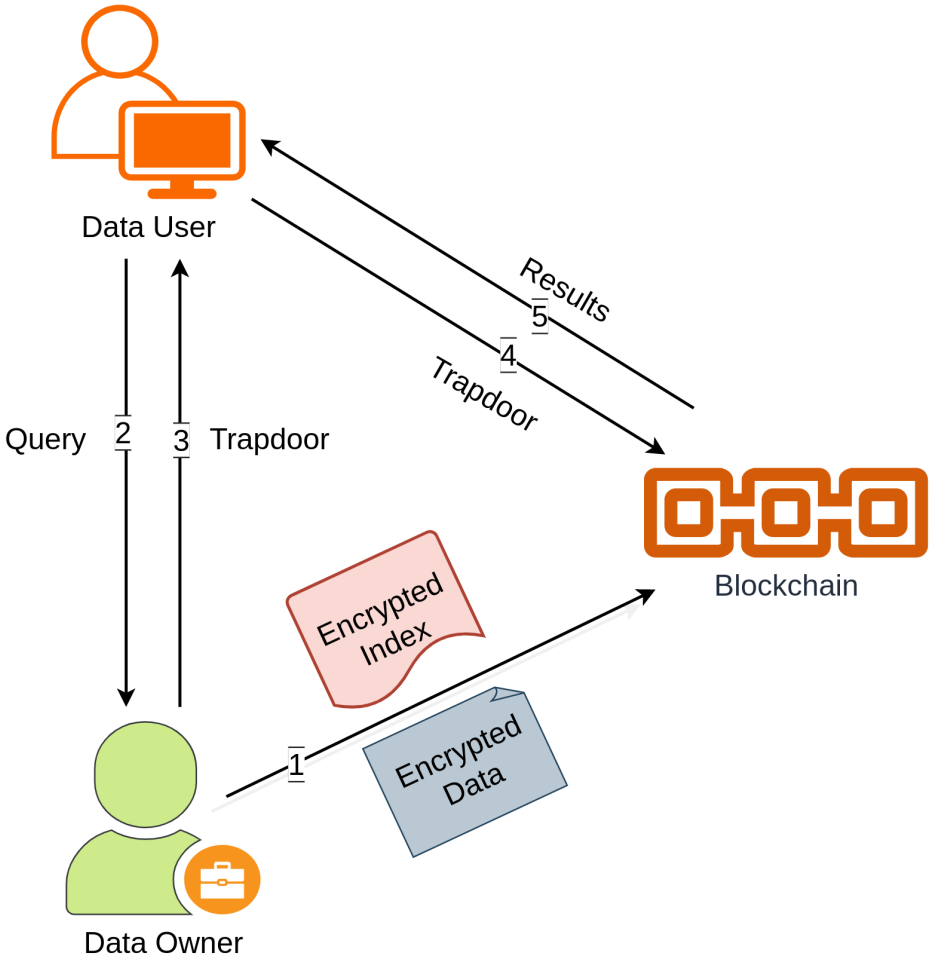

Figure 3.2: Searchable encryption with blockchain (Type 1)

3.3.2 Cryptographic primitives for SE

Based on the cryptographic primitives used, SE schemes can be categorized into two categories:

- *Searchable Symmetric Encryption (SSE):* This is the most widely studied form of searchable encryption pioneered by Song *et al.* [59] in 2000. It relies on symmetric key cryptography, hence the name. There are many variants of SSE such as sequential search based, inverted index based and tree based. However, the most popular one is inverted index based due its simplicity and sub-linear search time.

- *Public Key Encryption with Keyword Search (PEKS):* This searchable encryption technique allows keyword search over data encrypted using asymmetric cryptogra-

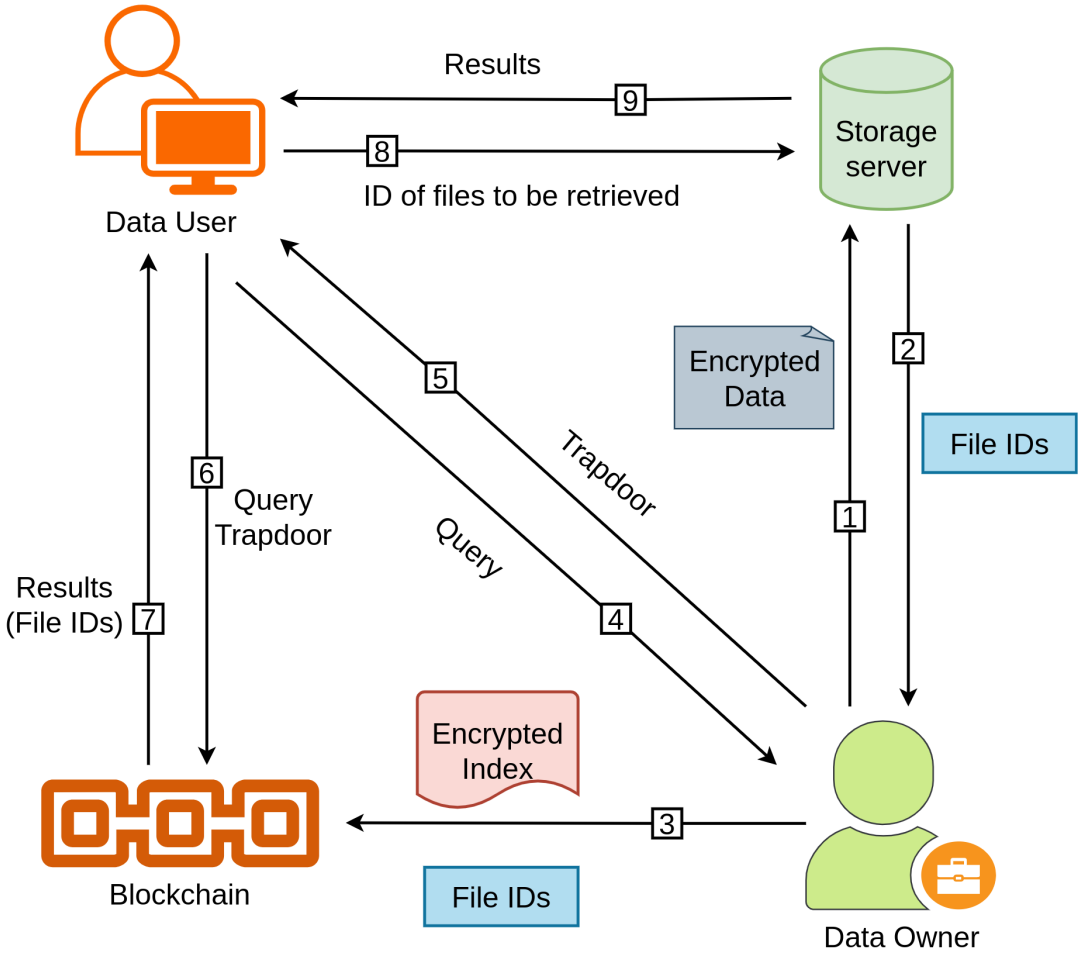

Figure 3.3: Searchable encryption with blockchain (Type 2)

phy. It was first introduced by Boneh *et al.* [85] to allow encrypted search over mails sent to a user. In usual case, multiple people can send mails to the user by encrypting them using his/her public key and the user will have to encrypt each of the mails to filter out relevant content. However, PEKS allows the user to filter his/her mails without actually decrypting them, thereby enabling the user to decrypt and read only those mails that he/she is interested in.

- *Attribute Based Encryption with Keyword Search (ABKS):* Attribute based encryption allows encryption of data in a manner such that only a person who has attributes that satisfy certain set of conditions (know as a policy) can decrypt it. For example, in an organization, attributes can be given to employees on the basis of their rank in the organization and the policy can be made in a way that higher

ranked employees can read the encrypted messages sent to lower ranked but not vice-versa. The first attribute based encryption scheme was proposed by Zhao *et al.* [118] in 2011.

- *Proxy Re-encryption with Keyword Search (PRES):* Proxy re-encryption is a technique which allows an entity, called proxy, to re-encrypt a message encrypted using public key of original recipient, into a message which can only be decrypted by a person chosen by the original recipient. In short, the original recipient delegates his decryption capability to another person on his behalf. To re-encrypt the original message, the proxy entity needs a re-encryption key and the public key of the delegate. The re-encryption is generated using the secret key of the original receiver but cannot be used to disclose the original receiver's secret key. An example where this is useful is when an official of an organization wants to delegate some of his messages to his/her secretary.

  This technique was combined with encrypted keyword search for the first time by Shao *et al.* [119]. In the use case presented by them, the proxy entity was a mail server which could re-encrypt the message intended for a recipient into message for a delegate.

3.3.3 Security models for SE

The first formal threat model for SE schemes was proposed by Goh [83]. The author proposed semantic security against chosen-keyword-attack through a threat model which he termed IND-CKA. This model aimed to cover the notion of security in which an adversary cannot deduce any partial information about the contents of a document from its index other than what it already knows from external sources. IND-CKA provided security only when the size of all the documents involved in creation of index is equal. This shortcoming was overcome by proposing a stronger model IND-CKA2 in which the size of the documents need not be equal. However, none of the two security models required trapdoors to be secure and therefore failed to model real world scenario for searchable encryption.

In 2006, Curtmola *et al.* [84] provided an alternative security model for SE, which since then has become a standard. They proposed two security models - 1) *Non-Adaptive Semantic Security* 2) *Adaptive Semantic Security.* The setting is that of a client and server, where the client generates a secure index and upload it to server along with the

encrypted files. The adversary is assumed to have the access to all the data that is leaked to the server during operations. Before understanding the security models in detail, we need to define some terminologies that will be used in to define them. We use informal definitions to describe them for the ease of understanding. To read the formal definitions we suggest the reader to go through the original paper [84]. Note that the security model proposed by Curtmola *et al.* [84] was designed for *SSE*, however, it can be modified for other SE techniques.

- *History:* It is defined as a tuple of collection of documents and set of queries that a user has made over them.

- *Access pattern:* For a given history access pattern is the set of document identifiers that were revealed by the queries present in the history.

- *Search pattern:* For a given history containing q queries, search pattern tells whether any two documents contain the same keyword or not. It is represented by using a binary matrix of size $q \times q$ where value at i^{th} row and j^{th} column is 1, if $w_i = w_j$; else, it is 0.

- *Trace:* The length of each document in the collection along with *search pattern* and *access pattern* for the trace of a given history.

- *Non-singular history:* A history is said to be non-singular if at least one history exists such that trace of both the histories is equal and such a history can be found in polynomial time. Trace is the information that we can allow to be leaked to

We now describe the security models presented by Curtmola *et al.*. In both the models, history must be non-singular.

- *Non-adaptive semantic security:* In this security model, the adversary is allowed to generate histories but only at once. Moreover, it is not allowed access to the secure index, document collection or the trapdoors of any of the queries until it has generated the histories. The scheme is considered secure if an adversary cannot distinguish between two histories that it has generated after getting access to secure index, trapdoors and encrypted documents. The authors have also provided a simulation based definition for non-adaptive security and proved it to be equivalent to the indistinguishability based definition.

- *Adaptive semantic security:* The adaptive security model is similar to the non-adaptive one except that the adversary can choose the histories adaptively. Specifically, the challenger randomly selects a bit b and asks the adversary to provide two documents D_0 and D_1. Following which, the challenger generates secure index I_b and provides it to the adversary. Next, the challenger asks the adversary to provide two keywords w_0 and w_1 and then returns trapdoor for the word w_b to the adversary. The adversary is allowed to repeat this query process polynomial number of times. After that, the adversary is challenged to output a bit b'. The scheme is considered secure if the adversary cannot distinguish between b and b' except with some negligible probability.

It is clear from the description above that adaptive semantic security is harder to achieve, it but guarantees greater security.

3.3.4 Attacks on SE schemes

The main security threats in searchable encryption scheme arise from the leakage that occurs during operations such as build and search. The first work to demonstrate a successful attack on SE schemes was presented by Islam *et al.* [89] in 2012. Their work exploited the leakage pattern and some prior knowledge to successfully disclose sensitive information. It demonstrated the need for a formalisation of leakages by SE schemes. Working in this direction Cash *et al.* [93] presented a formalisation of leakages in SE schemes They categorized SE schemes into four levels - L_1, L_2, L_3 and L_4, where L_1 denotes least amount of leakage and L_4 denotes highest.

Attacks on SE due to leakages can be divided into the following categories: [77]

- *Keyword Guessing Attack (KGA):* In this attack the adversary tries to decipher the token/trapdoor in order to recover the keyword that is being searched. This is possible because in practice the words in a document are from a low entropy message space and brute force can be used to encrypt each of the words in message space and compare with the received encrypted keyword. Based on the nature of the attacker, it can further be classified into the following two categories:
  - *Inside KGA:* In inside KGA [120], the cloud server on which the encrypted data and encrypted index is stored is itself malicious. Since the attacker has access to all the search queries, it is easy for them to launch the KGA attack.

  - *Outside KGA:* In outside KGA, the attacker is an external entity and has no relation with the user or cloud server. The attacker captures the data while it is being uploaded to the server and then later on tries to guess the keyword contained in a search query by comparing with word selected from the message space.

- *Inference attacks:* In this attack, the attacker tries to use the encrypted indexes to decipher keyword plaintext. Following are the commonly used methods.

  - *Frequency analysis attack:* In this attack the frequency of keywords present in the cipher-text and encrypted ranking information [121] in the outsourced data is analysed by the attacker to get the keyword plaintext.
  - *Sorting attack:* This type of attack is extremely effective for dense distribution. The attacker uses "keyword in the ciphertext to get plaintext from the encrypted indexes and tries to compute the encrypted ciphertext frequency."
  - *Counting attack:* This attack is based on the fact that a large fraction of keywords will match against a unique number of documents [93]. The adversary can then count the number of documents returned corresponding to a keyword and match it with number of documents matched by the query. The pattern of keywords across returned documents can be deducted if multiple keywords return same number of documents.

- *Access pattern attack:* Any information that can be used by an attacker to determine the frequency at which files are accessed or their association with the query is known as access pattern [80]. In this type of attack the attacker uses access pattern to guess the keywords from the trapdoor.

- *Search analysis attack:* Any information that can be used by the attacker to determine if random queries are related to keywords is known as search pattern[80]. In this type of attack the attacker usually tries to determine if a new trapdoor and previous trapdoor are derived from the same keyword.

3.3.5 Threat model

We follow the widely-accepted Dolev-Yao (DY) threat model [122] as it was used in security protocols for other networking environments. Under the DY-model, an adversary $\mathcal{A}$

will have an opportunity to tamper with the communicating data, such as reading, modifying, deleting or even inserting fake data during the communications of the transmitted messages among the entities in the network. Additionally, as discussed in Section 3.3.4, $\mathcal{A}$ can launch various attacks apart from traditional attacks (replay, man-in-the-middle and impersonation attacks), like KGA, inference, access pattern and search analysis attacks. Furthermore, based on the DY-model, the end point entities are not fully trusted. We assume that once the data in put into the blockchain, $\mathcal{A}$ can not tamper with the data inside blocks, because of inherent properties of the blockchain technology (decentralization, transparency and immutability). Finally, we assume that the cloud servers are treated as semi-trusted nodes in the network.

3.4 Existing schemes for PHR sharing

In this section, we describe various schemes for PHR sharing that leverage searchable encryption and blockchain.

Since computers became mainstream, researchers and industrialists have been trying to digitize the entire record keeping of health records and like every other digital information, PHRs have been prone to cyber attacks of various kinds especially due to the high value of the health data that they store. However, with the advent of blockchain many paths have opened up to tackle the shortcomings of the previous health data storage and sharing schemes. Application of searchable encryption over blockchain in healthcare is a new and active area of research. Most of the schemes using searchable encryption and blockchain, that have been proposed till date, have assumed their own model of the health care infrastructure. However, all of them follow one of the following described approaches. The first approach uses blockchain as a storage for PHRs as well as secure indexes. It is similar to the system model described in Figure 3.2. The second one uses blockchain only for storing secure indexes and metadata related to search and leave the storage of PHRs to a cloud server or a distributed database. It is similar to the system model described in figure 3.3.

Most of the schemes have three main entities - patient, hospital and blockchain network. A patient visits a hospital and registers himself to receive his unique id and security keys depending on the type of encryption used. The hospital creates an PHR for the patient based on the diagnosis provided by the doctor. The PHR is then encrypted and uploaded to the blockchain network after the required processing for generating secure

indexes is done. It is important to note that the doctor may access the medical history of the patient by asking for a token from the patient that allows him to query the blockchain network for the past medical records of the patient. This is possible because the PHRs are encrypted using searchable encryption scheme that allows authorized users to access the encrypted data. The exact details of the scheme vary depending on the assumptions that have been made about the network and storage infrastructure of the hospitals. Many of the searchable encryption scheme also use *Attribute Based Encryption (ABE)* [123] in order to have a fine-grained access control over the encrypted data. In those schemes access control structures such as AND-gate structure, tree-based structure and threshold structure are used.

3.4.1 Existing schemes

We now briefly describe the existing schemes for PHR sharing using SE and blockchain.

- **Zhang *et al.* [54] (2018):** The first specific solution to the PHR problem using blockchain and searchable encryption was proposed in 2018 by Zhang *et al.* [54]. It is based on "*Public Key Encryption with Keyword Search (PEKS)*". Their scheme, called *Blockchain based Personal Health Information Sharing (BSPP)*, targets efficient sharing of health information of patients for improvement in diagnosis. They have used public key encryption which allows other users to access the patient's data with the help of a search token. This scheme however allows only single keyword search. Both the PHRs and the meta-data related to searching is stored on blockchain. However, they have used two blockchain networks, one of which is private and the other is consortium. Private blockchain stores the PHRs in encrypted form, while consortium blockchain stores the secure indexes and other metadata of the encrypted PHRs.

  The proposed scheme contains three phases - 1) "*System Setup*", 2) "*Data Generation and Storage*", and 3) "*Data Search and Access*". During the system setup phase, the user and doctors executes key generation algorithm to generate their public private key pairs and *GlobalSetup* algorithm is run to generate system parameter $GP = (P_1, P_2, \hat{e}, H_0, H_1, H_2, H_3, H_4, H_5, H_6)$. P_1 and P_2 are generators of cyclic groups of prime order, $\hat{e}$ is bilinear mapping and $H_i, \forall i \in 0, ..., 6$, are hash functions. When a user i registers himself with the hospital k, he receives $\beta \in \{0,1\}^*$ through a secure channel. The hospital server also selects a doctor j for the user

and sends $\mu = H_1(\beta)$ to the doctor. The doctor uses this value to authorize the user when he/she visits him/her. The doctor j generates a health record $m \in \{0,1\}^*$ for the patient i and also selects $w \in \{0,1\}^*$ from the standard keyword set. m and w are encrypted by doctor using public key pk_i of the user. The encryption algorithm gives $c = (c_{i_0}, c_{i_1}, c_{i_2})$ as output, where c_{i_1} is the ciphertext for searchable keyword and c_{i_2} is the evidence required for proof of conformance algorithm proposed in this paper. c_{i_0} is stored in the hospital server and the hash value of c_{i_0} is stored in blockchain. To search the user generates two trapdoors, the first one is identity searching trapdoor T_d and second one is keyword trapdoor T_w for word w. T_d is required to search for the specific user and T_w is required to search for the specific word. Both are sent to the doctor.

- **Chen *et al.* [64] (2019):** In 2019, Chen *et al.* [64] proposed an Ethereum based searchable encryption scheme for PHRs. It uses *SSE* and allows queries with complex expressions. This is a bit different from the traditional searchable encryption scheme as the documents that are being encrypted are of MongoDB and the queries are not keywords, but instead they are expressions used to search the MongoDB. In this scheme, all the PHRs are scanned and the records that satisfy condition expression $X = \{X_1, X_2, ..., X_m\}$ (X_i is a conditional expression that can be used to search over a Mongo DB document) are extracted and id_{ij} is computed corresponding to document D_{ij} that satisfy X_i. Based on these extracted identifiers the index I is built which is uploaded to the blockchain and he encrypted documents are outsourced to a decentralized file system such IPFS.

  To search over an PHR a user authenticates himself/herself to the data owner and obtains a trapdoor/search-token. The user sends this trapdoor to the smart-contract on blockchain which returns the file identifiers on which the corresponding keyword is present. The user can then contact the storage server on which the encrypted files are stored and use the identifiers received from the smart contract to download them.

  The authors describe their goals as fairness, soundness and confidentiality. Fairness ensures that user will get accurate results if he pays for it. Soundness ensures that a dishonest party will be detected and will obtain no rewards. Confidentiality ensures that the PHRs are secure.

- **Niu *et al.* [114] (2019):** In 2019, Niu *et al.* [114] proposed a scheme using

Attribute Based Encryption using Keyword Search (ABKS) and blockchain. It is based on permissioned blockchain model and uses private key encryption and supports multiple keyword search. The scheme consists of three phases - 1) "*System Setup*", 2) "*Data Generation and Storage*", and 3) "*Data Search and Access*".

In the system setup phase, the system parameters $PP = (p, e, g_1, g_2, g_2^{\alpha}, g_2^{\beta}, g_1^{\frac{\beta}{\alpha}}, G_1, G_2, H_1, H_2, H_3, H_4)$ are generated along with master secret key $msk = (\alpha, \beta)$ Here, G_1 and G_2 are multiplicative cyclic groups with generators p, g_1 and g_2 are two generators of G_1, $e : G_1 \times G_1 \rightarrow G_2$ is an admissible bilinear map. $\alpha, \beta \in Z_p^*$ are randomly selected by the system and H_i is a hash function for $i \in 1, 2, 3, 4$.

In data generation phase, the patient P visits hospital and is randomly assigned $h \in Z_p^*$ through a secure channel and also assigned a doctor D. Next, $\mu = H_3(h)$ is generated and reserved by the server. The patient P chooses an access control structure Γ which is in the form of a tree and runs $Share(\Gamma, h)$ for each leaf node to obtain secret value $h_v(0)$ for the leaf node v and calculates $A_v = g_2^{h_v(0)}$, $B_v = g_2^{H_1(s_v)h_v(0)}$ for each leaf node attribute s_v. These values are sent to the doctor D, which runs an encryption algorithm to compute ciphertext C and keyword index I. The encryption algorithm $Enc(PP, \Gamma, W, M) \rightarrow (I, C)$ takes public parameters PP, access control structure Γ of the patient, a keyword set $W = w_1, w_2, ..., w_m$ of the shared PHRs and PHRs M as input and returns the keyword index I and ciphertext C as output. The doctor D then uploads C and I to the hospital database server.

In data search and access phase, data users construct the trapdoor of keywords $W' = \{w'_1, w'_2, ..., w'_n\}$ where $n \leq m$. The trapdoor algorithm $Trapdoor(PP, W', sk) \rightarrow T_{W'}$ takes the public parameters, search keywords set W' and secret key sk as input and outputs the trapdoor $T_{W'}$. Next, the participants in the permissoned blockchain run the search algorithm $Search(I, T_{W'})$ which takes keyword index I and trapdoor $T_{W'}$ and outputs ciphertext C. Finally the user upon receiving C decrypts the ciphertext using $Decrypt(PP, sk, C) \rightarrow M$ algorithm to obtain the message M.

- **Wang *et al.* [62] (2019):** In 2019 Wang *et al.* proposed a electronic health record sharing scheme using "Proxy Re-encryption with Keyword Search (PRES)" and blockchain. It consists of five entities - "*Data Owner(DO)*", "*Data provider (DP)*", "*Cloud Server (CS)*", "*Blockchain (BS)*" and "*Data Requester (DR)*". The proposed scheme has three layers - 1) "*Data Generation Layer*", 2) "*Data Storage*

Layer" and 3) "*Data Sharing Layer*".

A patient requires to create an account for consortium blockchain when visiting a hospital. During registration, a patient i receives an account address A_i and a private key from the consortium blockchain. The doctor is the data provider and he/she generates PHR m for the patient and extracts a list of keywords w_i from it. He/She then encrypts m using public key pk_i of the patient, private key x_k of the doctor and keyword w_i to generate cipher text C_m. The doctor also encrypts the keywords w_i using his/her public key X_k to generate the keyword ciphertext C_w. Next, $v_1 = (C_m||C_w||A_i)$ is uploaded to the cloud server and the receives file location F_i as the response from the server, which is then send to the patient. The doctor also sends data packet $v_2 = (C_w||A_i||C_k)$ to the blockchain. Here, C_k is doctor's signature for proof of conformance for the blockchain.

In this work, the search query is approved by the doctor instead of the patient unlike in most other works. When a doctor receives a keyword for search, he/she generates a trapdoor T_Q corresponding to it. Using the trapdoor, the data requester (DR) can search for the required PHR and account address A_i of the patient on the blockchain. Next DR sends $V_3 = (I_j||pk_j||X_k||A_j)$ to the data owner, that is, patient. Here, I_j is identity of data requester, pk_j is his/her public key, A_j is his/her account address, and X_k is doctor's public key. Upon receiving, the request from DR, patient authorizes it and sends the file location F_i and keyword set w_i to DR. Moreover, the patient also sends a re-encryption key rk to CS which carries out proxy re-encryption for the ciphertext required. Finally, the re-encrypted ciphertext is decrypted by the DR using his private key sk_j.

- **Sun *et al.* [113] (2020):** In 2020, Sun *et al.* [113] proposed a solution for the problem of secure storage and sharing of Personal Health Records(PHRs) using *Attribute based Encryption with Keyword Search (ABKS)*. In this work they have used *Inter Planetary File System (IPFS)* [117] to store the records. IPFS is a distributed file system that assigns a unique identifier to each file that is stored on it. The hash of the record is stored on Ethereum based blockchain network which ensures authenticity and integrity of the data. The encryption type used is private key, however, this scheme does not support multi-keyword search.

  The scheme consists of three parts - 1) *System Establishment*, 2) *Medical Data Generation and Storage*, and 3) *Data Search and Access*. In the system establish-

ment phase, the doctor generates public key PK for the system using algorithm $GlobalSetup(\lambda)$. Here λ is the security parameter. The algorithm selects $\alpha, \beta \in Z_p^*$ and a hash function $H : \{0,1\}^* \rightarrow G_0$ and computes $A = e(g_0, g_0)^\alpha, B = g_0^\beta$ where $e : G_0 \times G_0 \rightarrow G_1$ is a symmetric bilinear map, G_0, G_1 are multiplicative cyclic groups and g_0 is generator of G_0. Further $\xi \in Z_p^*$ is selected randomly and the search private key is set as $sk_u = \{\xi\}$. Additionally, each patient $i \in U$ selects $y_i \in Z_p^*$ randomly and sets its as his/her private key sk_i and announces public key as $pk_i = g_0^{y_i}$. Similarly, each doctor $j \in D$ selects $x_j \in Z_p^*$ randomly and sets it as his/her private key and announces public key as $pk_j = g_0^{x_j}$.

During data generation and storage phase, for each patient i, a value $\varphi_i \in Z_p^*$ is randomly selected by hospital k and is then sent to patient i securely along with assigning a doctor j for the patient. The server calculates $\mu_i = H(\varphi_i)$ which is used to authorize the patient. The doctor then generates PHR m and a set of keywords W_m associated with m. He/She then encrypts m and W_m based on the access policy negotiated with the patient and generates ciphertext CT and encrypted index $index$.

When a data user wants access to the medical records of the patient, he/she sends a request to access the records along with his Ethereum public key to the doctor. The doctor assigns some attributes to the user and adds his account address to the list of authorized users after verifying his/her identity.

- **Tang *et al.* [65] (2021):** Tang [65] proposed another blockchain based *SSE* scheme for medical record sharing in 2021. First, the authors have presented a solution in which the entire medical record of the patient is retrieved on searching i.e without fine grained access control and then have presented a modification to the original scheme to allow fine-grained access over the medical records of the patient. Their scheme consists of four entities - 1) *hospital collection*, 2) *patient collection*, 3) *Blockchain system* and 4) *authority*. It has the following phases - 1) *Index-building*, 2) *Data retrieval*, 3) *Integrity verification*, 4) *Data addition*, and 5) *Data deletion*. The data addition and deletion property make this scheme dynamic in nature. It also provides forward security which ensures that previous search queries do not leak any information about the PHRs that are being added later.

  During system initialization, smart contracts are deployed and the entities involved generate their keys and security parameters. During index building phase, a patient O_i visits hospital H_j where a doctor generates the medical records denoted

by $PHR_{i,j,c}$ for the patient. Here, c denotes the serial number of medical record. Hospital H_j summarizes the records for the patient O_i when they reach a certain number and generate a secret key using global identifier of the patient O_i. This secret key is used to build encrypted search index and verification index which are uploaded to the blockchain network using smart contract. During data retrieval phase, a doctor employed at another hospital H_k tries to retrieve medical history of O_i for diagnosis. O_i gives the required data for trapdoor generation to the hospital which then retrieves the PHR identifiers from the encrypted index using smart contract. The hospital then requests the corresponding hospital to send the encrypted medical records. The received records are decrypted and checked for integrity and accuracy using the verification index. After diagnosis is over new PHRs after generated by H_k and corresponding entries are added to the encrypted search index and verification index. The patient O_i also updates his locally stored information so that a new trapdoor could be generated in the future. This is important for forward security of the scheme.

- **Wang *et al.* [61] (2022):** This scheme proposed by Wang *et al.* [61] is based on *ABKS*. They have combined consortium blockchain with a distributed file system IPFS for secure storage and sharing of PHRs. They have used attribute base encryption to allow a fine-grained access control mechanism over the PHRs and have used zero-knowledge proof for storage evidence of PHRs. The proposed scheme consists of five main entities - 1) *Data Source (DS)*, 2) *Data Owner (DO)*, 3) *Data User (DU)*, 4) *Blockchain Client (BC)* and a 5) *Blockchain Network (BN)*. Their main goal are - 1) *Data Security*, 2) *Secure Search*, 3) *Collusion-Resistant Privacy Preservation* and 4) *Personalized Access Control*.

  The scheme proposed by the authors consists of the following phases 1) *System Initialization*, 2) *Data Generation* and 3) *Data Sharing* phases. Data generation further has the following sub-phases - 1) *Cipher-text Generation*, 2) *Storage Proof Generation*, 3) *Keyword Ciphertext Generation*, 4) *Symmetric Key Cipher-text Generation* and 5) *Smart Contract Generation*. During System Initialization phase, system parameters such as master public key and master secret key are setup. In Cipher-text Generation phase, DO encrypts the PHR using symmetric key encryption. This is followed by storage proof generation where, DO uploads encrypted PHR to IPFS and receives a unique identifier which is used as input for zero-knowledge proof to

compute storage proof. In verify and consensus phase, the storage proof is sent as a transaction to the blockchain network which uses *Practical Byzantine Fault Tolerance (PBFT)* as the consensus algorithm. Next, DO chooses a set of keywords from the PHR and a set of attributes consisting of personal information of the owner and uses them to generate keyword ciphertext. This is followed by encryption of symmetric key used to generate the ciphertext of PHR. Finally, DO writes symmetric key ciphertext, identifier of PHR ciphertext and required incentive fee into a smart contract.

In the data sharing phase, DU requests for attribute key from attribute agency by providing his/her attributes. This attribute key can be used to generate trapdoor for a set of keywords that the DU wants to search for. This is done by sending the attribute key and trapdoor to the smart contract which validates both and returns PHR identifier and the symmetric key to decrypt the PHR. DU can then download the encrypted PHR from IPFS using the identifier and decrypt it using the key received from smart contract.

3.4.2 Comparative study

In this section, a comparison of the schemes discussed above has been provided. Generic parameters have been chosen for comparison of features that are offered by the schemes considered above. It consists of type of cryptographic primitive used, type of blockchain model used, type of keyword search and dynamism. There are four types of cryptographic primitives as discussed in Section 3.3.2, namely - "*Searchable Symmetric Encryption (SSE)*, *Public key Encryption with Keyword Search (PEKS)*, *Attribute based Encryption with Keyword Search (ABKS)* and *Proxy Re-encryption with Keyword Search (PRES)*". We refer *Type-1* and *Type-2* for the system model as shown in Figures 3.2 and 3.3, respectively. The type of blockchain model can be public, private or consortium as described in Section 1.2.2. The type of keyword search refers to the type of queries allowed in the search. Single keyword search and multiple keyword search are two possible types of keyword searches. However, some schemes also use database query expressions such as that of Chen *et al.* [64] instead of the standard keyword search.

Table 3.3 shows a comparative study on various features supported by the schemes under consideration as per the criteria discussed above.

Table 3.3: Features comparison of various blockchain-based EHR sharing schemes

Year	**Scheme**	**Cryptographic Primitive**	**Blockchain Type**	**Keyword Search Type**	**Dynamic**	**System Model**
2018	Zhang *et al.* [54]	PEKS	Private + Consortium	Single	No	Type-1
2019	Chen *et al.* [64]	SSE	Public	MongoDB query expressions	No	Type-2
2019	Niu *et al.* [114]	ABKS	Private	Multiple	No	Type-2
2019	Wang *et al.* [62]	PRES	Consortium	Single	No	Type-2
2020	Sun *et al.* [113]	ABKS	Public	Single	No	Type-2
2021	Tang *et al.* [65]	SSE	Public	Single (for fine-grained search)	Yes	Type-2
2022	Wang *et al.* [61]	ABKS	Consortium	Single	No	Type-2

3.5 Summary

In this chapter, we described searchable encryption and blockchain which are currently leading technologies. We explained how search over encrypted data can be achieved and provided an understanding of the blockchain technology which has been steadily growing in use over the past decade. Additionally, we provided a description of PHRs and the benefits of using them. Furthermore, we described how we can combine SE and blockchain together, and use them to create secure systems that are capable of secure storage and sharing of PHRs. We also discussed the latest works that have used SE and blockchain for PHRs storage and sharing. Finally, we conclude that both of these technologies show great promise for the future advancements in healthcare technologies.

Chapter 4

Efficient PHR Sharing in IoMT using Searchable Symmetric Encryption, Blockchain and IPFS

In this chapter we propose and elaborate an efficient solution for sharing and storage of personal health records using searchable encryption, blockchain and IPFS. We first discuss the system model which consists of the details of various entities involved and the inter-links between them. We then describe the protocol between these entities followed by formal security proofs and test-bed experimental results.

4.1 System model

In this section, we discuss the system architecture and threat model used in the development of the proposed scheme.

The system model proposed in this work consists of various patients, hospitals, a blockchain network consisting of peer nodes maintained by hospitals, and an IPFS network that is accessible through the Internet. Figure 4.1 shows a diagrammatic representation of our system model. It shows two hospital instances designated as *Hospital 1* and *Hospital 2*. Each hospital contains a *hospital server*, a *Blockchain node*, several *doctor devices* and several *medical devices*. In practice there will be more number of hospitals; however, for clarity we have shown only two hospitals. Further, there are two patients designated as

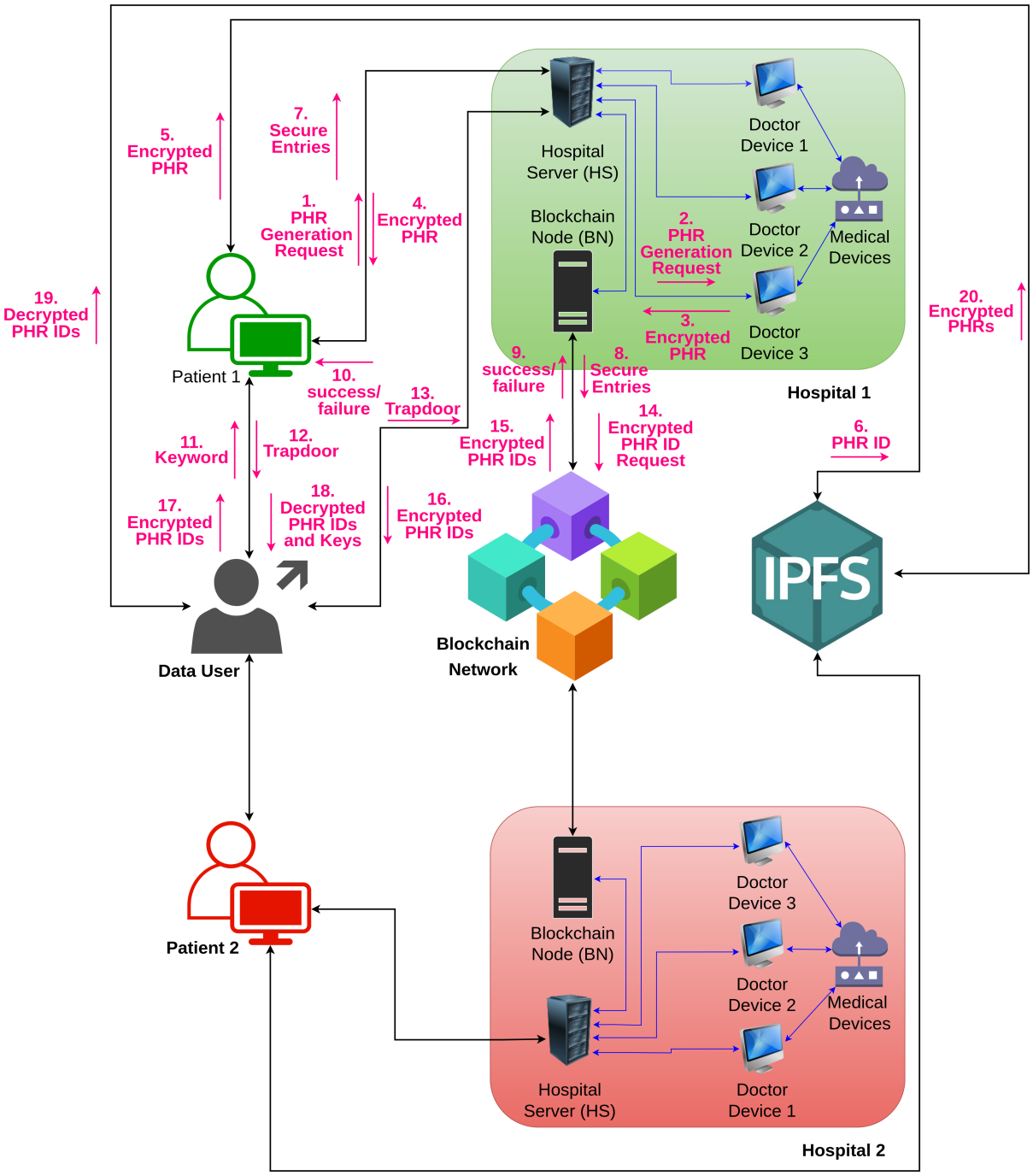

Figure 4.1: System model

Patient 1 and *Patient 2*. In practice, there will be many patients and each patient can visit any of the hospitals for diagnosis but for simplicity, we have shown only two patients. *Patient 1* visits *Hospital 1* and *Patient 2* visits *Hospital 2*. A patient is alternatively called

data owner. Further, we have an instance of *data user* in the model. A data user can interact with any of the hospitals and data owners. The exact nature of this interaction will become clear in subsequent sections.

Figure 4.1 shows the blockchain and IPFS networks. Each hospital is also a part of the consortium blockchain network. It is used to store meta-data such as the encrypted index. IPFS, on the other hand, is independent of the hospitals and used to store the actual PHRs in encrypted form. We can say that the PHRs are stored in an off-chain structure that is decentralized; thereby providing more reliability.

Apart from presenting the different entities involved in our system model, Figure 4.1 also describes the network model of our system. The connector lines in the figure represent communication channels between the entities. Bidirectional arrows denote the fact that the channels are bidirectional. The channels shown using black lines are built over the Internet while the channels denoted using blue lines are built over local network of the entity in which they are contained. For instance, the communication channel between *Patient 1* and *Hospital Server* is shown using a black line, while that of *Doctor Device 1* and *Hospital Server* is shown using a blue line.

We have shown the various messages that are passed between the entities in the system during its operation. These messages are shown using the color pink and numbered in a sequential manner so as to describe the flow of information in the system. Note that the messages are not shown for Patient 2 and Hospital 2 as they would be similar to that of Patient 1 and Hospital 1. Additionally, each entity in the system generates its own pair of elliptic curve keys which are then used for secure communication with other entities using "Elliptic Curve Integrated Encryption Scheme (ECIES)".

We describe each of the entities present in the system model in detail.

- *Patient* (P)*:* A patient is a person who visits a hospital for medical diagnosis and is also called the data owner.

- *Hospital Server* (HS)*:* We define hospital to be a medical institution that provides services to a patient. Each hospital that is part of our system should also maintain a server (denoted by HS) that handles all the requests from patients and a blockchain node (BN) that connects the hospital to the blockchain network.

- *Doctor* (D)*:* Every instance of a hospital has several doctors associated with it and each doctor is provided a device through which he/she can access the hospital's IT infrastructure.

- *Data User* (DU)*:* Any entity that wants to make use of the data uploaded into the system by data owners is called a data user. A patient itself can also become a data user. Another common instance of a data user is a doctor who requires a patient's medical history for diagnosis. Moreover, insurance companies can also be categorized as data users.

- *Blockchain Node* (BN)*:* In our scheme, every hospital in the ecosystem should be a part of the consortium blockchain network, and thus should maintain a blockchain node that takes part in the consensus mechanism.

- *IPFS:* It is used by patients to store their PHRs (in encrypted form). It can be accessed through the Internet.

- *Medical devices:* Each hospital has a set of medical devices to diagnose patients and are connected to the local network of the hospital. A few examples of such devices are "Electrocardiogram (ECG) device", "Electroencephalogram (EEG) device" and "Magnetic Resonance Imaging (MRI) device", etc. A doctor can use these devices to generate data to be included in the PHR.

Table 4.1: Notations and their definitions

Notation	**Definition**
$h(\cdot)$	Collision-resistant hash function
MK	A patient's master key
TS	Timestamp
$E_{PK}(\cdot)$	Probabilistic ECIES encryption using public key PK
$E_k^{SK}(\cdot)$	Probabilistic symmetric encryption using key k
$X[k]$	Accessing element of dictionary X using key k
X_i	Accessing i^{th} element of list X
$\|\|$	Concatenation
U	Universal set for keywords

4.2 Proposed scheme

Till now, we have discussed about individual entities in our system. We now move ahead and describe the interactions between these entities which forms the core of our proposed scheme. First, we define the various components essential for understanding the scheme and then provide its overview followed by a detailed description.

The proposed scheme provides dynamic updates by allowing addition of new PHRs without having to rebuild the index. It is important to note that our scheme does not provide the functionality to update the contents of an individual PHR. It only allows addition of new PHRs dynamically, i.e., without having to rebuild the index. PHR records are generated by a doctor; therefore, a data owner does not need the facility to themselves update an individual PHR. Moreover, the literary works have only focused on addition of new PHRs and not on updating an individual PHR. Here, we also focus on the same. After PHR addition, a data user can search for PHRs containing specific keywords with consent from the data owner. For ease of understanding, we have provided a list of notations used to describe our scheme, in Table 4.1.

4.2.1 Definitions

Definition 4.1 (PHR)**.** *A PHR is a text file containing the diagnosis report generated by a doctor.*

Definition 4.2 (Keyword)**.** *A keyword is a sequence of characters that is delimited by a special character such as a blank space.*

Definition 4.3 (Secure Entries Dictionary)**.** *It is a dictionary denoted by Δ and is used to store the entries of the encrypted search index.*

More details about Δ are presented in Section 4.2.3.

Definition 4.4 (Trapdoor)**.** *A trapdoor is a special search token that is issued by data owner to a data user when the latter wants to search for the PHRs containing a given keyword.*

The exact structure of a trapdoor is provided in Section 4.2.3.

Definition 4.5 (PHR Sharing Scheme)**.** *We define our scheme as a tuple* $\Sigma = (KwExt, SEGen, PHRAdd, TrapGen, Search, RetId)$ *where the algorithms are described as follows:*

- $W \leftarrow KwExt(PHR)$*: KwExt algorithm takes a PHR as input and returns a list* W *of unique keywords present in the PHR.*
- $\Delta \leftarrow SEGen(W)$*: SEGen algorithm takes a set of unique keywords* W *as input and returns a secure entries dictionary* Δ *as the output.*
- $s \leftarrow PHRAdd(\Delta)$*: PHRAdd algorithm takes a secure entries dictionary as input and stores its content on the blockchain MR-tree. It returns the status of the operation denoted by* s *which can be either* success *or* failure*.*
- $t \leftarrow TrapGen(w)$*: TrapGen algorithm takes a keyword* w *as input and returns the corresponding trapdoor* t *for search.*
- $\Gamma \leftarrow Search(t)$*: Search algorithm takes a trapdoor* t *as input and returns a list* Γ *of encrypted identifiers of PHRs containing the keyword corresponding to* t*.*
- $\Upsilon \leftarrow RetId(\Gamma)$*: RetId algorithm takes search result* Γ *as the input and returns the actual PHR identifiers* Υ*.*

4.2.2 Overview

Our scheme consists of the following phases: 1) *setup*, 2) *patient registration*, 3) *PHR generation and addition*, and 4) *keyword search.*

We first give an overview of the scheme in this section and then describe each phase in detail.

In the *setup* phase, the entities select security parameters, such as public-private key pairs and other secret values. This phase is executed only once during the system initialization. The second phase, *patient registration*, is executed by a patient whenever he/she visits a hospital for the first time. After completion of this phase, the patient

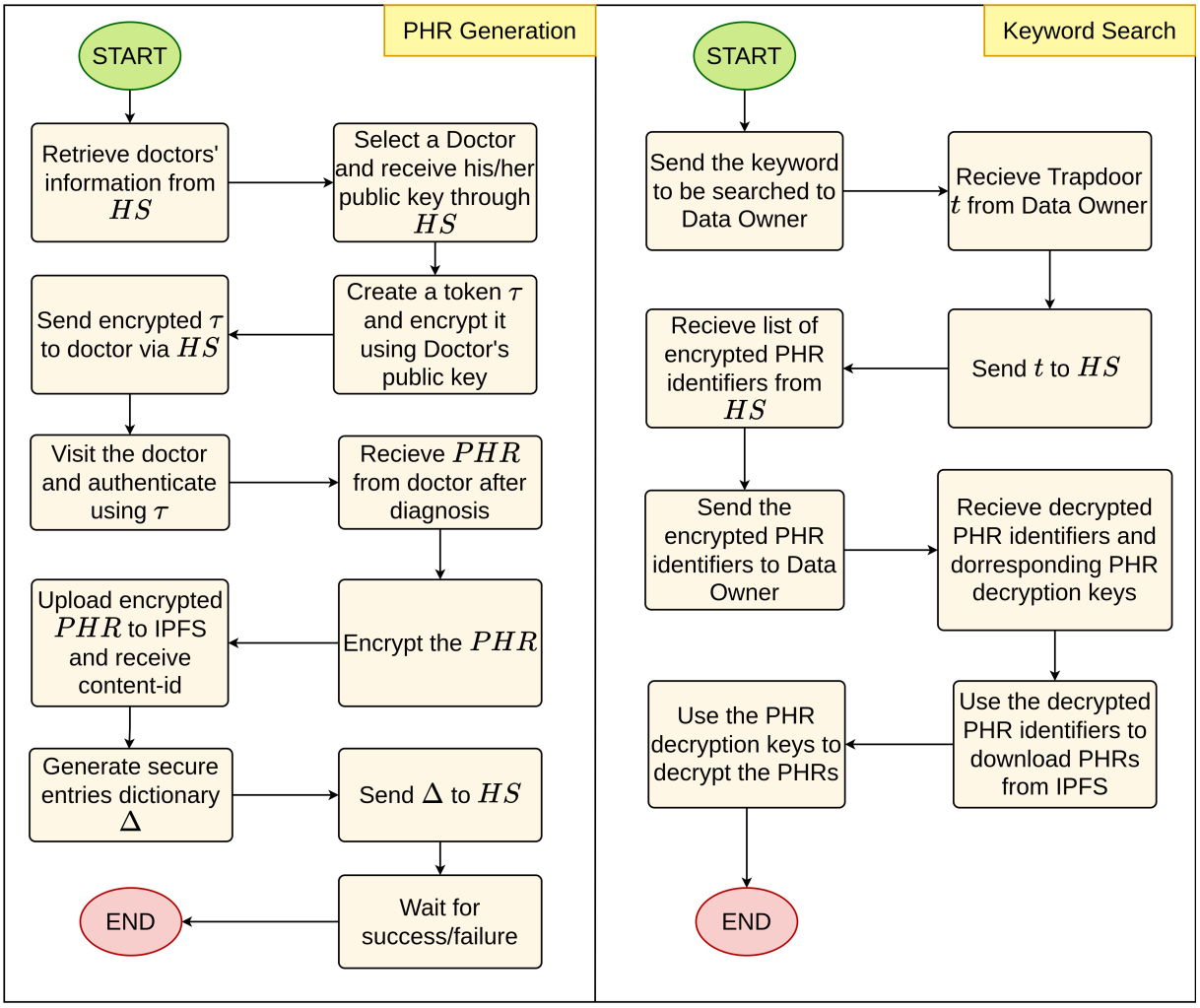

Figure 4.2: Flowcharts for PHR generation and keyword search

receives a globally unique identifier and password. The patient uses the same identifier and password when visiting any other hospital later.

The third phase, *PHR generation and addition*, is initiated by a patient when he/she requires health diagnosis. The patient retrieves the list of doctors from the hospital and selects a doctor for treatment. He/she then sends a request to the chosen doctor via HS to retrieve the doctor's public key. Finally, he/she sends a token to the doctor via HS. The token is later used to authenticate the patient when he/she visits the doctor.

Upon connecting with the doctor, the patient is authenticated using the token. The doctor diagnoses him/her only if the token provided by the patient matches the one it received earlier. Diagnosis involves collecting data from the IoMT smart devices and

giving medical prescriptions based on the patient's condition. The doctor then constructs a PHR of the patient containing all this information and sends it to the patient securely [1]. Upon receiving the PHR, the patient runs $KwExt$ algorithm to retrieve unique keywords from the PHR. The patient then uploads the PHR to IPFS and receives a unique content-id for it denoted by id_{phr}. Finally, the patient runs $SEGen$ algorithm and sends the output Δ from the algorithm, to the hospital server. The hospital server sends this data to its BN, which submits it as an add transaction to the blockchain network. Finally, this transaction is processed and added to blockchain by the peers. A flowchart of PHR generation process from the perspective of a patient can be seen in Figure 4.2.

The fourth and final phase, *keyword search*, starts when a data user wants to search for PHR containing a particular keyword. He/she requests for a trapdoor corresponding to the keyword to be searched from the patient. The data user sends this trapdoor to the hospital server, which, in turn, runs the search algorithm and returns search results containing encrypted ids of the PHRs containing the given keyword along with some possible dummy entries. The dummy entries are a means to provide security to our scheme (see Section 4.2.3 for details). The data user can get actual PHR ids and decryption keys by sending the search results to the patient/data owner. After obtaining the decrypted PHR id from the patient, the user can then directly query IPFS and download the encrypted PHRs. Since the PHRs are stored in encrypted form, the data user first needs to decrypt them using the key sent by patient along with the actual PHR id. A flowchart of this process from the perspective of data user is also shown in Figure 4.2.

4.2.3 Phases

We now describe the phases mentioned above, in detail.

1) Setup

In this phase, the entities generate security parameters such as public-private key pairs. It has the following sub-phases.

- *Global Setup:* A non-singular elliptic curve $E_p(a, b)$ of the form: $y^2 = x^3 + ax + b \pmod p$ is chosen where p is a large prime, and $a, b \in Z_p = \{0,\ 1,\ 2,\ \cdots, p-1\}$ such that $4a^3 + 27b^2 \neq 0 \pmod p$ is satisfied. Additionally, a generator G is chosen

[1] Channels are secured using ECIES encryption

from $E_p(a, b)$ whose order will be as large as p. Next, a "collision-resistant one-way cryptographic hash function, $h(\cdot)$" with output length l is chosen which acts as a random oracle $\mathcal{O}$. A security parameter n is also chosen. The keyword space which contains all the valid keywords that can be present in a PHR is denoted by U. Finally, for ECIES encryption, ECDH key exchange and probabilistic AES-OCB3 encryption [75] are chosen as the underlying components.

- *Patient Setup:* Each patient (device) randomly selects an integer $k_p \in Z_p^*$ as its private key, where $Z_p^* = \{1, 2, \cdots, p-1\}$. The corresponding public key is computed as $K_p = k_p \cdot G$ where $(\cdot)$ represents elliptic curve scalar multiplication, that is, $k_p \cdot G = G + G + \cdots + G$, k_p times. In addition to public-private key pair, each patient randomly selects a master key $MK \in \{0, 1\}^n$. Further, each patient device maintains a dictionary which we term as "Current Head Dictionary (CHD)". It takes a keyword (sequence of characters) as its key and stores a PHR id as the corresponding value. Patient device also maintains a reverse look-up dictionary RLD which maps encrypted PHR ids to actual PHR ids and the PHR decryption keys. The significance of these dictionaries will become clear in the PHR generation (see Section 4.2.3).

- *Hospital Setup:* In a similar manner to a patient device, HS randomly selects its private key $k_h \in Z_p^*$ and computes public key as $K_h = k_h \cdot G$. It is important to note that blockchain node, BN, will also use this key because HS and BN are maintained by the same entity (i.e., hospital). HS also maintains a dictionary that we term as "User Dictionary (UD)" which takes the patient's unique id as the key and the patient's public key as the value. It is used to retrieve the public key of the patient for encryption when HS wants to send some data to the patient.

- *Doctor Setup:* A doctor device is assumed to be maintained by each doctor. Doctor device randomly selects its private key $k_d \in Z_p^*$ and computes the public key as $K_d = k_d \cdot G$. Each doctor has a unique identity id_{D_i} for the hospital where he/she is employed.

- *Data User Setup:* Any data user that wants to interact with the system must select a private key k_u randomly from Z_p^* and compute the corresponding public key as $K_u = k_u \cdot G$.

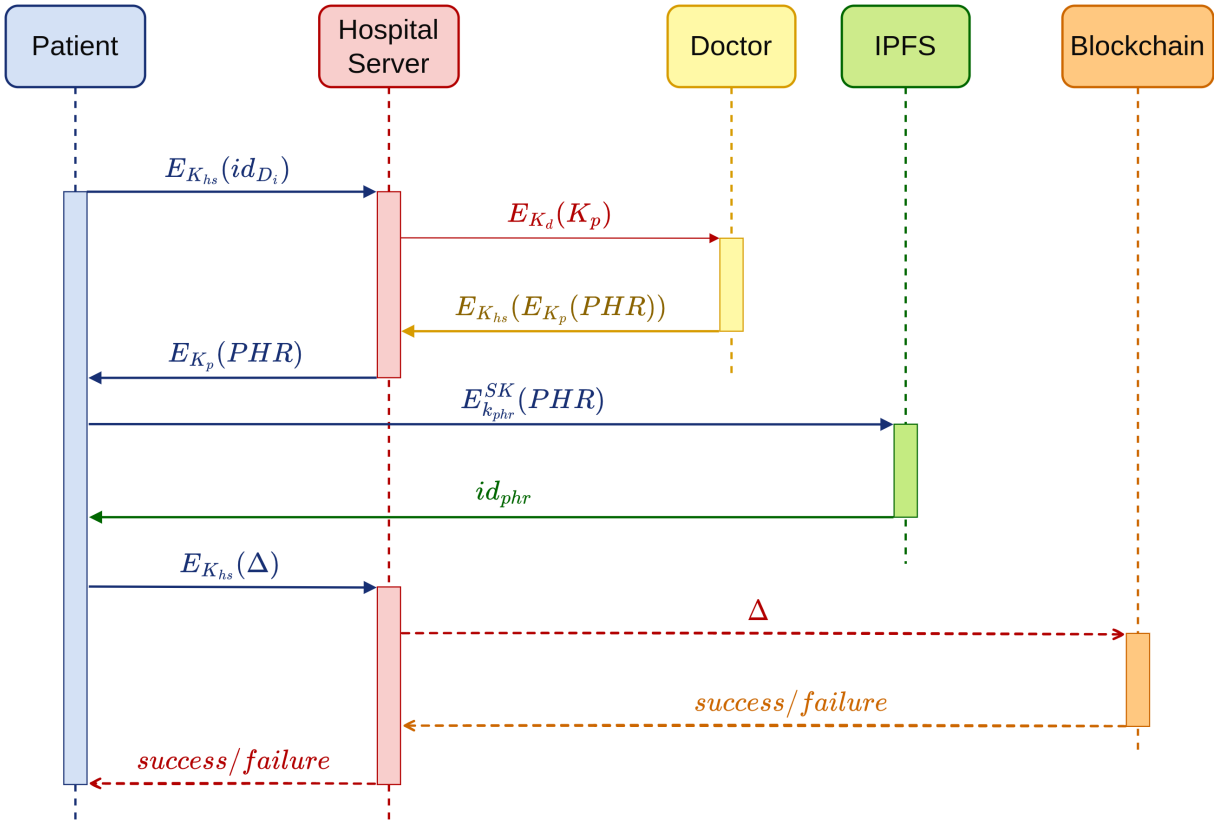

Figure 4.3: PHR generation sequence diagram

2) Patient registration

Suppose a patient, P wants to visit a hospital H. He/she registers himself/herself by providing the public key K_p to HS and then selecting a unique id and password. This id and password combination is used by the patient to authenticate himself or herself while communicating with HS. Upon completion of registration, the patient receives the public key K_h of the hospital server HS. Note that there are other ways the patients can receive K_h (for example, the hospital can list its public key on its website and use a certificate authority for its validation). However, to avoid providing unnecessary details and to simplify the scheme, we have assumed that the hospital server will simply send its public key back to the patients during registration. We have used similar ways to share public keys throughout our scheme, but if required they can listed on a public platform. It does not have any impact on the security of the scheme.

3) PHR generation

An overview of the PHR generation process is given in Figure 4.3. We describe the entire process in detail in this section.

- *Initiation:* To initiate PHR generation, a patient P requests an updated list of doctors from the HS. The patient P generates $\tau \in \{0,1\}^*$ as a token and then selects a doctor D from the list and sends a request to D through HS to get D's public key. P then encrypts τ using public key of D and sends it to him/her via HS.

Algorithm 1: Keyword extraction

Input: PHR

Initialize a set ks

for *each keyword* w *in PHR* **do**

  insert(ks, w)

end

Convert ks to list l

return l

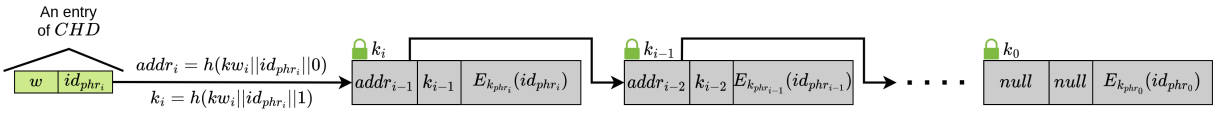

Figure 4.4: A single entry of the conceptual encrypted index

Later, when the patient interacts with D [2], he/she sends the token τ to the doctor for verification. If it matches the token previously sent by the patient to the doctor, then the doctor diagnoses the patient. If required, the doctor collects data from the medical devices and generates a PHR. Doctor device then encrypts the PHR using public key K_p of the patient. This encrypted PHR is finally sent to the patient P.

- *Secure Entries Generation:* P upon receiving the encrypted PHR, decrypts it using his/her private key and then runs Algorithm 2, which is an implementation of

[2]Patient has the option to interact in-person or remotely depending on the need.

SEGen, over the PHR to get secure entries dictionary Δ. Algorithm 2 is explained below in detail.

- *PHR upload:* Algorithm 2 first generates the PHR encryption key $k_{phr} = h(MK \| TS)$, where MK is P's master key and TS is the current time stamp. Next, the algorithm executes Algorithm 1, an implementation of $KwExt$, to generate the list of unique keywords, $W = \{w_0, w_2, \cdots, w_{n-1}\}$, present in the PHR. It then encrypts the PHR using the generated key and uploads it to IPFS, which returns a content id, denoted by id_{phr}. This id is encrypted as $id'_{phr} = E^{SK}_{k_{phr}}(id_{phr})$ to store it in the encrypted index. The encryption is necessary to prevent leakage of information to hospital. During implementation the encryption function $E^{SK}(\cdot)$ is replaced by AES-OCB3.

  P finally stores the actual PHR id id_{phr} and PHR decryption key k_{phr} in the reverse lookup dictionary RLD by using id'_{phr} as the key. RLD is used to map an encrypted PHR id to an actual PHR id and the decryption key during keyword search (more details are provided in Section 4.2.3).

- *Noise Addition:* Instead of using the original set of keywords W to generate the secure entries, we first add some noise to it by merging Q with W. Q is a subset of the keyword space U. The new keyword set is denoted by V. This addition of noise is required in order to prevent attacks against our scheme (see Section 4.3.4).

- *Encrypted Search Index:* Before going any further into explanation of Algorithm 2, we need to understand the structure of the encrypted search index and how it is stored in the Blockchain. The encrypted search index can be considered as a dictionary with a unique keyword w as the key and a linked list as the corresponding value. The linked list stores the encrypted identifiers of the PHRs that contain keyword w. This encrypted search index is partially stored on the patient device and partially in the blockchain's Merkle-Radix (MR) tree. Specifically, the key-set of this dictionary is stored on the patient's device in the form of CHD dictionary; while the value-set, which is essentially a set of linked lists, is stored in the MR-tree. This division is essential for the security of the search index, because generation of trapdoor is dependent on the key-set. If we store it on the blockchain, anyone can publicly access it and generate the trapdoor corresponding to a keyword. This would defeat the purpose of having a trapdoor.

Algorithm 2: Secure entries generation

Input: PHR, Q

$TS \leftarrow$ Current Timestamp

$k_{phr} \leftarrow h(MK \parallel TS)$

$W \leftarrow \text{KwExt}(PHR)$

Upload $E^{SK}_{k_{phr}}(PHR)$ to IPFS and receive id_{phr}

$id'_{phr} \leftarrow E^{SK}_{k_{phr}}(id_{phr})$

$RLD[id'_{phr}] = (id_{phr}, k_{phr})$

$V \leftarrow Q \cup W$

for $0 \leq i < |V|$ **do**

- $kw \leftarrow h(MK \parallel V_i)$
- $addr \leftarrow h(kw \parallel id_{phr} \parallel 0)$
- $k \leftarrow h(kw \parallel id_{phr} \parallel 1)$
- **if** $CHD[V_i] = null$ **then**
  - $addr^{prev} \leftarrow null$
  - $k^{prev} \leftarrow null$
- **else**
  - $id^{prev}_{phr} \leftarrow CHD[V_i]$
  - $addr^{prev} \leftarrow h(kw \parallel id^{prev}_{phr} \parallel 0)$
  - $k^{prev} \leftarrow h(kw \parallel id^{prev}_{phr} \parallel 1)$
- **end**
- **if** $V_i \in W$ **then**
  - $c \leftarrow E^{SK}_{k}(addr^{prev}, k^{prev}, id'_{phr})$
- **else**
  - $r \xleftarrow{\$} \{0,1\}^{l_{id_{phr}}}$
  - $c \leftarrow E^{SK}_{k}(addr^{prev}, k^{prev}, E^{SK}_{k_{phr}}(r))$
- **end**
- $\Delta[addr] \leftarrow c$
- $CHD[V_i] \leftarrow id_{phr}$

end

return Δ

Figure 4.4 shows one entry of this conceptual dictionary. Recall that CHD is a dictionary maintained by the client device. Specifically, $CHD[w]$ stores the id of the PHR that contains w in it and was added most recently to the system. This PHR identifier forms the head node of the linked list associated with keyword w along with some dummy nodes (discussed in part - Node Generation). As mentioned earlier, the linked-list corresponding to key w is used to store the identifiers of all the PHRs that contain w. Therefore, there is a linked list corresponding to every unique keyword contained in the set of PHRs of a patient. Each node of a linked list is stored individually in the MR-tree of the blockchain and has a unique MR-tree address. A node is linked to the next one by storing the MR-tree address of the next node in the current node. We have used a linked-list over other data structures, such as an array, because by design it ensures forward security (details are provided in proof of Theorem 4.2).

- *Node Generation:* As discussed earlier, each keyword present in V should have a corresponding linked list. The next part of the algorithm generates the nodes for these linked lists. For each keyword $w \in V$ a new node for the linked list corresponding to w will be created. This node is added to the front of the linked list and becomes the new head node. Adding the node from front, rather than back, is required to ensure forward security. In i^{th} iteration of the *for* loop present in the algorithm, a new head node for the linked list corresponding to keyword V_i is created. Specifically, for $0 \leq i < |V|$, the algorithm generates kw, $addr$ and k as follows: $kw = h(MK \parallel V_i)$; $addr = h(kw_i \parallel id_{phr} \parallel 0)$; $k = h(kw \parallel id_{phr} \parallel 1)$. Here, kw is an intermediate expression, $addr$ is the MR-tree address where the node will be stored and k is the key which will be used to encrypt the contents of the node.

  Before constructing the new head node, the address and decryption key of the current head node are required. Therefore, the algorithm obtains the PHR id that forms the current head node from CHD as $id_{phr}^{prev} = CHD[V_i]$. This id is then used to generate the MR-tree address of the new head node and the decryption key for the contents of new head node as follows:

$$addr^{prev} = h(kw \parallel id_{phr}^{prev} \parallel 0)$$
$$k^{prev} = h(kw \parallel id_{phr}^{prev} \parallel 1).$$

  If CHD does not contain an entry for V_i, then $addr^{prev}$ and k^{prev} are set to *null* value.

Finally, the algorithm calculates c. If the keyword V_i is present in the PHR, then c is computed as follows.

$$c = E_k^{SK}(addr^{prev}, k^{prev}, id'_{phr}).$$

Otherwise, the value of id'_{phr} is replaced by encryption of a random value r of same length as id_{phr} as follows.

$$c = E_k^{SK}(addr^{prev}, k^{prev}, E_{k_{phr}}^{SK}(r)).$$

The condition above is necessary to find the correct PHR identifiers during keyword search. Details are present in Keyword Search Section 4.2.3.

Now, c is the new head node of the linked list corresponding to the keyword V_i. It contains three values: a) address of the previous head, b) key of the previous head, and c) encrypted PHR id or encrypted random value. Also note that, in Figure 4.4 we have not shown dummy nodes for the sake of simplicity. However, in practice there will be dummy nodes mixed with the valid nodes.

All new head nodes denoted by $\{c_i \mid i \in \{0, 1, 2, \ldots, |V|-1\}\}$ are stored in dictionary Δ by using $addr_i$ as the key. The algorithm returns Δ as its output.

- *Uploading to Blockchain:* After generating Δ Patient P sends it to HS which forwards it to the "Blockchain Node (BN)". BN runs Algorithm 3 over Δ which

Algorithm 3: PHR addition

Input: Δ

for $(addr, val) \in \Delta$ **do**

    state.set($addr$, val)

end

Signal completion of the event to HS

retrieves each key values pair $(addr, c)$ and stores c at address $addr$ of the MR-tree, and records this transaction in a block. Upon completion of this operation, BN notifies HS about the successful execution of the transaction, which in turn shares the success response with P.

3) Keyword search

An overview of the keyword search process is given in Figure 4.5. This section describes the search process in detail.

Algorithm 4: Trapdoor generation

Input: w

$kw \leftarrow h(MK \parallel w)$

if $CHD[w] = null$ **then**

  $id_{phr}^{head} \leftarrow null$

else

  $id_{phr}^{head} \leftarrow CHD[w]$

end

$addr \leftarrow h(kw \parallel id_{phr}^{head} \parallel 0)$

$key \leftarrow h(kw \parallel id_{phr}^{head} \parallel 1)$

$t \leftarrow addr \parallel key$

return t

- *Trapdoor generation:* Suppose a data user (DU) wants to search for a PHR of a patient P that contains a specific keyword w. DU sends the keyword w to P to receive a trapdoor t corresponding to w. For this, P runs Algorithm 4 which calculates $kw = h(MK \parallel w)$. It then retrieves the PHR id stored in head node of the linked list corresponding to w as follows:

$$id_{phr}^{head} = CHD[w]$$

  The algorithm then calculates the address and key of the head node as follows:

$$addr = h(kw \parallel id_{phr}^{head} \parallel 0)$$
$$k = h(kw \parallel id_{phr}^{head} \parallel 1)$$

  $addr$ is the MR-tree address of the head node of the linked list corresponding to w and k is the symmetric key using which contents of the head node were encrypted. The algorithm returns $t = addr \parallel k$ as its output which is then sent by P as the trapdoor to DU.

Algorithm 5: Keyword search

Input: t

$addr, key \leftarrow$ Parse t

Initialize a list Γ

while *True* **do**

  Send a request to the blockchain node to get data at $addr$

  Store the response in res

  if $res.code = ok$ **then**

    $C \leftarrow res.content$

    $addr^{prev}, k^{prev}, x \leftarrow D_{key}(C)$

    $Append(\Gamma, x)$

    if $addr^{prev} = null$ **then**

      break

    end

    $addr \leftarrow addr^{prev}$

    $key \leftarrow k^{prev}$

  else

    return *(not-found, null)*

  end

end

$c \leftarrow E_{K_h}(\Gamma)$

return (ok, c)

- *Search: DU* then sends the trapdoor t and its own public key to HS after encrypting it using K_h[3]. HS then executes Algorithm 5 which directly queries the state of the blockchain's MR-tree using REST API. Specifically, using the trapdoor t, HS obtains the address $addr$ and key k required to obtain the head node of the linked list. HS sends a request using the blockchain's REST API to retrieve data at the address $addr$ of the MR-tree. The response of the request is $C = E_k^{SK}(addr^{prev}, k^{prev}, x)$,

[3] K_h can be obtained directly from hospital

Algorithm 6: Retrieve identifiers

Input: Γ

Initialize list Υ

for $0 \leq i < |\Gamma|$ **do**

  if RLD *contains* Γ_i **then**

    $id_{phr}, k_{phr} \leftarrow RLD[\Gamma_i]$

    $append(\Upsilon, (id_{phr}, k_{phr}))$

  end

end

return Υ

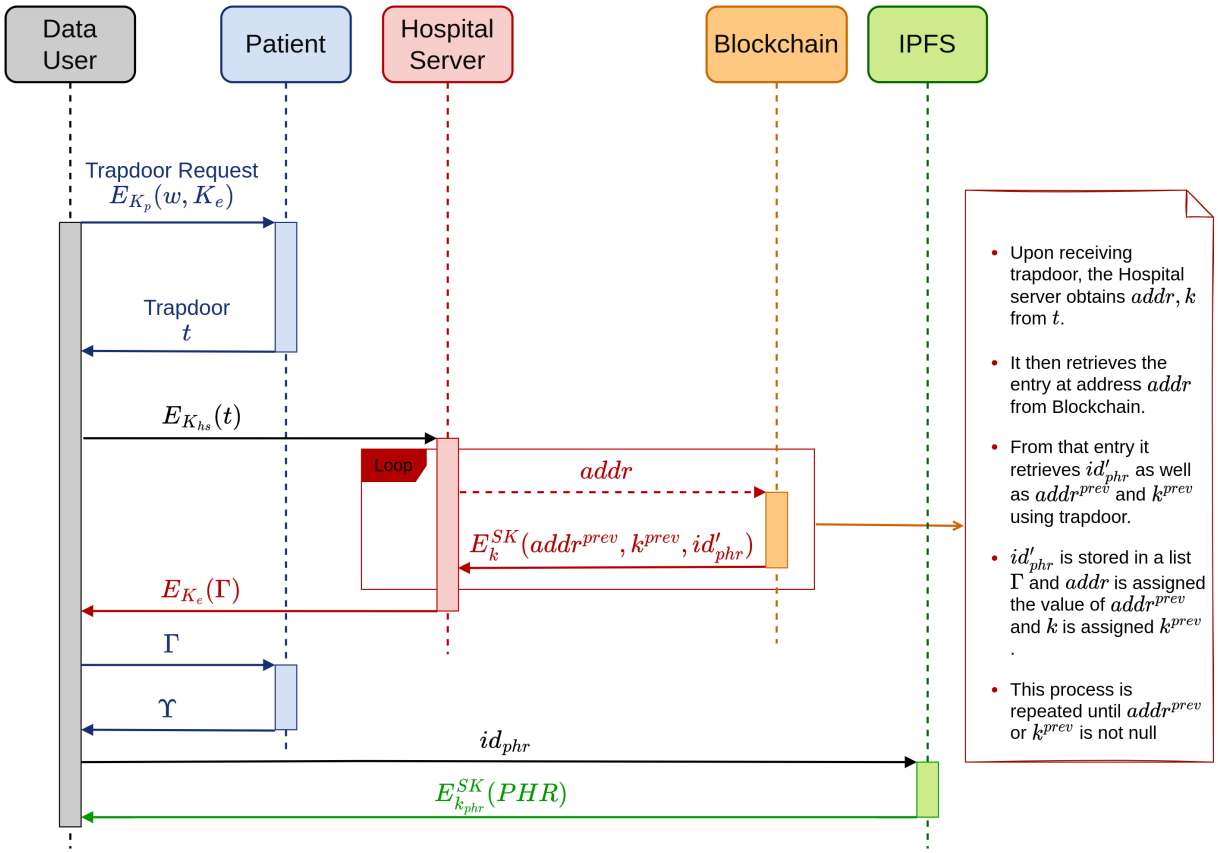

Figure 4.5: Search process

where x can be either id'_{phr} or $E_{k_{phr}}^{SK}(r)$. x is appended to a list Γ by the HS. $addr^{prev}$

and k^{prev} are then used as the new values to obtain the previous node. This process continues until $addr^{prev}$ or k^{prev} is not null. After completion of Algorithm 5, HS sends Γ to DU after encrypting it using DU's public key. Note that, for simplicity, we assume that HTTP protocol is being used for communication. Therefore, we have used some of the HTTP status codes like "ok" and "not-found" in the algorithm to denote the status of responses.

- *PHR identifier retrieval:* DU sends Γ to P in order to obtain the actual PHR identifiers. P runs Algorithm 6 to retrieve the actual identifiers of PHRs that contain w. The algorithm uses reverse lookup dictionary RLD for this purpose. If Γ_i denotes an element in Γ, then P checks if an entry corresponding to Γ_i exists in RLD. If it exists, then actual PHR identifier and decryption key are retrieved as $id_{phr}, k_{phr} \leftarrow RLD[\Gamma_i]$ and are appended to list Υ along with the PHR decryption key k_{phr}. Finally, the list Υ is sent to DU which can then retrieve the encrypted PHR from IPFS and decrypt them.

4.3 Security analysis

For proving the security of our scheme, we use the real-ideal simulation paradigm [124] that is used in most of the previous searchable encryption works [84], [88], [97] and prove that our scheme provides *semantic security* against an *adaptive adversary* in a semi-honest setting. First, we define the threat model under which our system is secure. This is followed by definition of information leakage of our scheme in the form of leakage functions. We then show that a simulator that takes these leakage functions as its inputs is indistinguishable from the real protocol for the adversary. This is followed by a proof that our scheme provides *forward secrecy* and is also *verifiable*.

4.3.1 Threat model

We evaluate security our proposed scheme by considering hospital as the adversary. All the important data in the system flows through the hospital, therefore, if we can show that our scheme is secure against an adversary with capabilities of hospital, we can be sure that it is secure against any adversary with lesser capabilities.

We consider a hospital to be *honest-but-curious*. It executes the protocol honestly,

but can try to learn about the information it is unauthorized to access. It can do so by using the data leaked to the hospital server or the Blockchain node during execution of the protocol. We collectively represent this leakage in the form of leakage-functions which are described in Section 4.3.2.

We assume a doctor to be trusted, because he/she is the entity which generates the contents of the PHRs. Moreover, by securing the communication between any two entities in the system using ECIES encryption, we can ensure that a doctor is not impersonated by an adversary.

4.3.2 Security definitions

Definition 4.6 (Search Pattern)**.** *A search pattern describes whether a keyword previously searched for is searched again. Formally, if m is the number of search queries that have been made, the search pattern is a matrix Z of dimensions $m \times m$ such that the value $Z_{i,j}$ is 1 if $w_i = w_j$; otherwise, it is 0. Here, w_i is the keyword searched during i^{th} query of a data user.*

Definition 4.7 (Leakage)**.** *The leakage for Σ is defined as a tuple $L = (L_{SEGen}, L_{PHRAdd}, L_{Search})$ where L_{SEGen}, L_{PHRAdd} and L_{Search} are leakages that occur during the secure entry generation, addition and search, respectively. These leakages are inherent in all the efficient searchable encryption schemes present till now. Note that, our scheme does not leak anything during the setup phase, because there is no interaction between the entities during this phase. Each entity performs the setup offline, and as a result, there is no possibility of any leakage except through physical means.*

We define L_{SEGen}, L_{PHRAdd} and L_{Search} as follows:

- **L_{SEGen}**: *SEGen takes a PHR as input and gives secure entries dictionary Δ as the output. In SEGen, the length (or size) of the PHR is leaked during its upload to the IPFS. Since the PHR is encrypted, no other information is leaked but the length can still be figured out from the ciphertext. We denote the length of the PHR by m. Thus, L_{SEGen} is defined as a function that takes a PHR as input and outputs m.*

- $\mathbf{L_{PHRAdd}}$*: During the addition of a PHR, the only information that is leaked by our scheme is the number of entries in* Δ*. This is because we give only* Δ *to the* HS*. Since the key is output of a hash function and the value is output of a probabilistic encryption function,* HS *can only figure out its size (the number of entries in* Δ*) but not the actual content. Thus,* L_{PHRAdd} *takes a set of these keywords* W *as input and outputs* α*.*

- $\mathbf{L_{Search}}$*: During search corresponding to a keyword* w*, the information that is leaked to the server is the number of files containing the keyword* w*, denoted by* β*, and search pattern* Z*. Formally,* L_{Search} *takes a keyword* w *as input and outputs* (β, Z)*.*

Definition 4.8 (Adaptive Semantic Security for Σ)**.** *We define security of our scheme similar to [97]. Let* $n \in \mathbf{N}$ *be a security parameter and* $m = poly(n)$ *be a positive integer, where* $\mathbf{N}$ *is the set of natural numbers. For simplicity of proof we assume that patient acts as a data owner as well as a data user. This does not affect the security of the scheme as data user is considered trusted.*

Consider the experiments $Ideal_{\mathcal{F},\mathcal{S},\Psi}$ *and* $Real_{\Sigma_{\mathcal{F}},A,\Psi}$ *executed between stateful adversary* A *and stateful simulator* $\mathcal{S}$ *using leakage functions* L_{PHRAdd} *and* L_{Search}*.*

- $Ideal_{\mathcal{F},\mathcal{S},\Psi}$ **:** *Environment* Ψ *asks the data owner to run SEGen, Add or Search operation by giving it the required information. For SEGen,* Ψ *gives a PHR as input. For Add,* Ψ *gives secure-entries to be added to blockchain as input and for Search it gives a keyword. Data owner sends the operation to ideal functionality* $\mathcal{F}$*.* $\mathcal{F}$ *gives the corresponding leakage to* $\mathcal{S}$ *which returns either abort or continue to* $\mathcal{F}$*. For SEGen, it returns abort or secure entries dictionary. For Add, it returns abort or "done" and for Search, it returns abort or list of encrypted PHR/dummy identifiers.* Ψ *observes the output and finally returns a bit* b *as the output.*

- $Real_{\Sigma_{\mathcal{F}},A,\Psi}$ **:** *Environment* Ψ *asks the data owner to run* $SEGen$*, Add or Search operation and provides the required information. For* $SEGen$*, it provides a PHR.*

For Add, it provides a secure entries dictionary and for Search it provides a keyword. Data Owner executes the corresponding operation in presence of a real world adversary A. Data Owner outputs either abort or secure entries dictionary for SEGen. For Add it outputs abort or 'Done' and for Search, it outputs abort or list of encrypted PHR/dummy identifiers . Ψ observes the data owner's output and finally returns a bit b as the output.

We say that a scheme Σ_F emulates an ideal functionality $\mathcal{F}$ if for all real world PPT adversary A, there exists a PPT simulator $\mathcal{S}$ such that for all polynomial time environments Ψ there exists a negligible function $negl(n)$ such that:

$$|Pr[Real_{\Sigma_{\mathcal{F}},A,\Psi}(n) = 1] - Pr[Ideal_{\mathcal{F},\mathcal{S},\Psi}(n) = 1]| \leq negl(n)$$

Definition 4.9 (Forward Security)**.** *A scheme $\Sigma = (KwExt, SEGen, PHRAdd, TrapGen, Search, RetId)$ is forward secure [90], if the leakage that occurs due to previous Add and Search operations do not reveal any partial information about a newly added PHR except what is revealed by leakage L.*

Definition 4.10 (Verifiability)**.** *A scheme $\Sigma = (KwExt, SEGen, PHRAdd, TrapGen, Search, RetId)$ is said to be verifiable, if the results returned to the user through the search operation are valid.*

4.3.3 Security claims

1) Adaptive-semantic security

The correctness of Theorem 4.1 ensures that our scheme is confidential against a PPT adversary in semi-honest model. Once proven secure against hospital, we can be sure that our scheme is also secure against any other adversary that does not have access to all the leaked information.

Theorem 4.1. *If h is a hash function and $E^{SK}(\cdot)$ is a CPA secure encryption scheme, then our PHR sharing scheme Σ is secure based on Definition 4.8 under random oracle model.*

Proof. To prove that the real game and ideal game are indistinguishable by any PPT distinguisher, we provide an implementation of PPT Simulator $\mathcal{S}$ that takes only the information provided by leakage functions as input to simulate patient behaviour in a way that is indistinguishable from real patient. $\mathcal{S}$ initializes two dictionaries *LenDict* and *ResDict*. *LenDict* uses a keyword as the key and stores an integer as the value. *ResDict* uses a keyword as the key and stores a list as the value.

For SEGen, $\mathcal{S}$ returns a secure-entries dictionary by executing the S_{SEGen} routine described in Figure 4.6. The routine takes $\alpha = |V|$ as the input. $\mathcal{S}$ obtains V using $W \cup Q$, where W can be easily obtained from the PHR and Q is known publicly. The routine simply creates α number of random entries for the dictionary Δ and returns it as the output. The key is a randomly chosen from $\{0,1\}^l$ and the value is probabilistic encryption of $0^{(l+n+l_{id_{phr}})}$. Here, l is the output length of hash function, n is the security parameter and $l_{id_{phr}}$ is the length of PHR identifier. The key and value of each entry are indistinguishable based on random oracle model and the CPA security of E^{SK}, respectively. Thus, Ψ is unable to distinguish between the real and ideal games using the *SEGen* operation.

During Add $\mathcal{S}$ simply returns 'Done' because the addition of a PHR in real game does not return anything besides abort or 'Done'. During Search $\mathcal{S}$ must return a list of encrypted PHR/dummy identifiers. For this $\mathcal{S}$ executes the routine described in Fig 4.6. It takes an input β which is the length of the list of encrypted PHR/dummy identifiers. This information is provided to $\mathcal{S}$ by the leakage function L_{Search}. $\mathcal{S}$ uses *LenDict* to keep track of the length of the list that is to be returned as output when keyword w is searched. When w is searched for the first time, and the leakage corresponding to it is β then a new entry (w, β) is created in *LenDict*. *ResDict* also gets a new entry with w as the key and a list containing β encryptions of $0^{l_{id_{phr}}}$ ($l_{id_{phr}}$ is the length of the PHR identifier, which depends on the hash function used by IPFS). This list is returned as the output. Each entry of the list is indistinguishable based on the CPA-Security of E^{SK}. When w is searched again, the same list is returned. Now, for the case when w is searched after addition of one or more PHR, the simulator just adds the number of entries equal to the difference between β and $LenDict[w]$. $LenDict[w]$ is then updated to β and $ResDict[w]$

$\mathbf{S}_{\mathbf{SEGen}}(\alpha)$

Initialize dictionary Δ
for $1 \leq j \leq \alpha$
  $addr \xleftarrow{\$} \{0,1\}^l$
  $k \xleftarrow{\$} \{0,1\}^l$
  $\Delta[addr] = E_k^{SK}(0^{(l+n+l_{id_{phr}})})$
endfor
return Δ

$\mathcal{S}_{\mathbf{Search}}(\beta, \mathbf{w})$

if $LenDict[w] = null$
  $LenDict[w] \leftarrow \beta$
  Initialize $ResDict[w]$ as an empty list
  for $0 \leq i < \beta$
    $k \leftarrow \{0,1\}^n$
    $append(ResDict[w], E_k^{SK}(0^{l_{id_{phr}}}))$
  endfor
endif
if $\beta \neq LenDict[w]$
  for $0 \leq i < (\beta - LenDict[w])$
    $append(ResDict[w], E_k^{SK}(0^{l_{id_{phr}}}))$
  endfor
  $LenDict[w] \leftarrow \beta$
endif
return $ResDict[w]$

Figure 4.6: Implementation of simulator

is returned. Thus, in every case, Ψ gets consistent results and it is therefore unable to distinguish the ideal game from real game. □

2) Forward security

It is an important property of any SE scheme. It ensures that the data leaked during the operations of the proposed scheme do not reveal any partial information about the PHRs that will be added in the future. For example, if an adversary knows that a PHR contains a certain keyword, he/she must not be able to determine if a PHR that is added later contains the same keyword or not.

Theorem 4.2. *Our scheme Σ is forward-secure in semi-honest model under the random oracle model.*

Proof. When a new PHR is added, the address is computed as $h(kw \parallel id_{phr} \parallel 0)$ (see Section 4.2.3). Here, id_{phr} is computed by taking the hash of the contents of the PHR and therefore is unique for each PHR. Thus, even for a keyword that occurs multiple times in different PHRs, the address generated will always be a random value output by the random oracle h. Further, new nodes are added to the head of the linked list in the existing encrypted index. This ensures that upon addition of a new PHR id to the same list does not require any modification in the encrypted index. Even in an extreme scenario where an adversary has once queried all the keywords whose entries are in the encrypted index[4], the adversary will not be able to figure out which keywords are present in the newly added PHR. Hence, our scheme is forward secure. □

3) Verifiability

Verifiability refers to the fact that the results returned by the hospital to the user are correct. For example, an adversary can return only partial results to the data user in order to save computational cost. Out of total number of PHRs containing a given keyword, the server may return only a fraction of those. A data sharing scheme must be safeguarded against such types of attacks.

[4]This can happen when the adversary has some pre-knowledge about the patient's disease

Theorem 4.3. *Our scheme Σ is verifiable under the standard blockchain model.*

Proof. According to the standard blockchain model [73], a blockchain is trusted for correctness and availability, but not privacy. Since a hospital server is part of a consortium blockchain network and is honest but curious, we can be sure that once the hospital receives a search request, it will forward it to the blockchain node. The blockchain node will return the correct search results from the distributed ledger by using Algorithm 5 because it is a trusted for correctness under the standard blockchain model. Therefore, the results received by the user will always be guaranteed to be valid. □

4.3.4 Security against other attacks

1) Query recovery using file injection attack

In 2016, Zhang *et al.* [125] proposed a query recovery attack that could retrieve all the query made by the user. This devastating attack works on the schemes where the adversary is able to inject files of their own choice and the query pattern is leaked to the adversary during search. This attack is prominent in cases such as application of SE over emails where the adversary can just send an email containing keywords of their own choice to the data owner.

In our case, however, this attack is resisted by the use of dummy entries (see Algorithm 2). The attack relies on the knowledge of a subset of keyword space (denoted by Q) and the access pattern (the set of identifiers of files that contain the keyword being searched), however, our scheme hides the access pattern from the hospital by adding dummy entries from keyword space in the secure entries dictionary. Now, if the adversary somehow manages to inject a file with keywords of their own choice, then when a data user searches for a keyword w from Q, the search results (which consists of encrypted PHR/dummy identifiers) returned do not actually reveal if the PHR contains w or not. This is because the result returned can be a dummy encryption. The data user will have to query the data owner to get the actual results. This way the adversary won't be able to learn the access pattern better than guessing.

2) Content recovery using known keyword set attack

This is an attack specific to SE schemes that allows addition of single files to the system instead of adding as a batch. If the adversary knows a keyword set $W' \subset U$, which contains some or all of the keywords in the keyword set W of the PHR, then after addition of a PHR the adversary can immediately search for all the keywords in W' and determine the content of the PHR with high accuracy. This attack is very devastating if W' and W have large number of keywords in common.

Our scheme provides protection against this attack by merging a large enough set of keywords $Q \subseteq U$ with the set of keywords W present in the PHR (see Algorithm 2). After getting the search results, the data user has to request data owner to execute Algorithm 6 over the search results to get the actual PHR identifiers. This ensures that an adversary cannot gets the actual PHR identifiers unless he/she impersonates the data user. The impersonation will be possible only if the adversary can steal the private key of the data user. Therefore, just as in previous attack, the adversary will not be able to determine if the search results obtained are valid or just some dummy entries. Thus, our scheme is secure against PHR recovery using known keyword set attack.

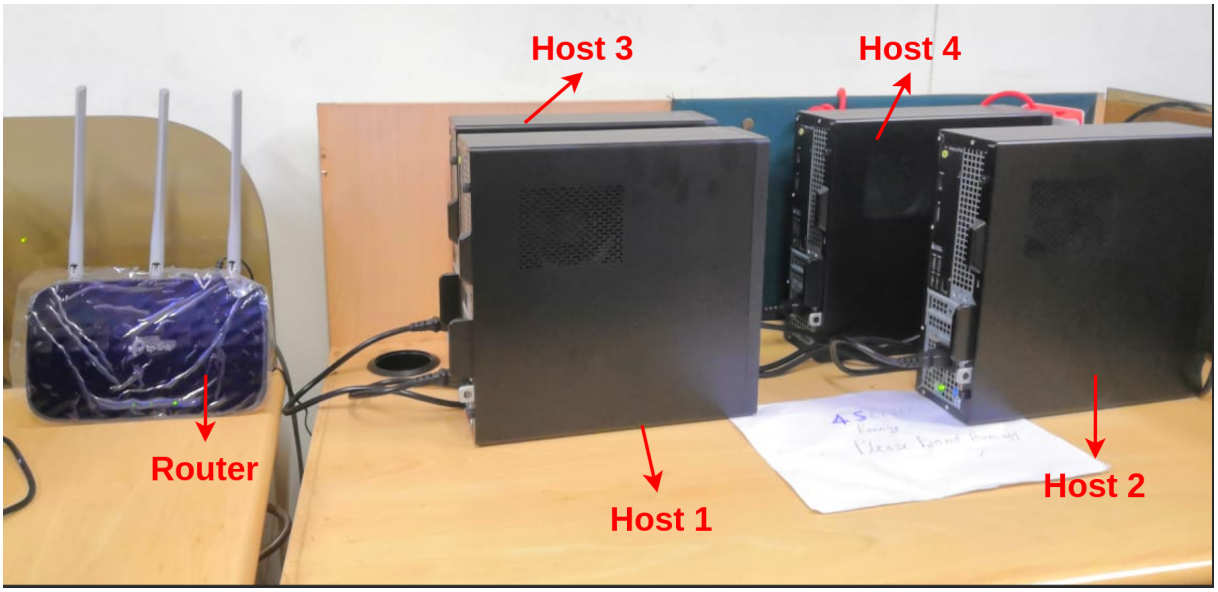

Figure 4.7: Experimental setup 1

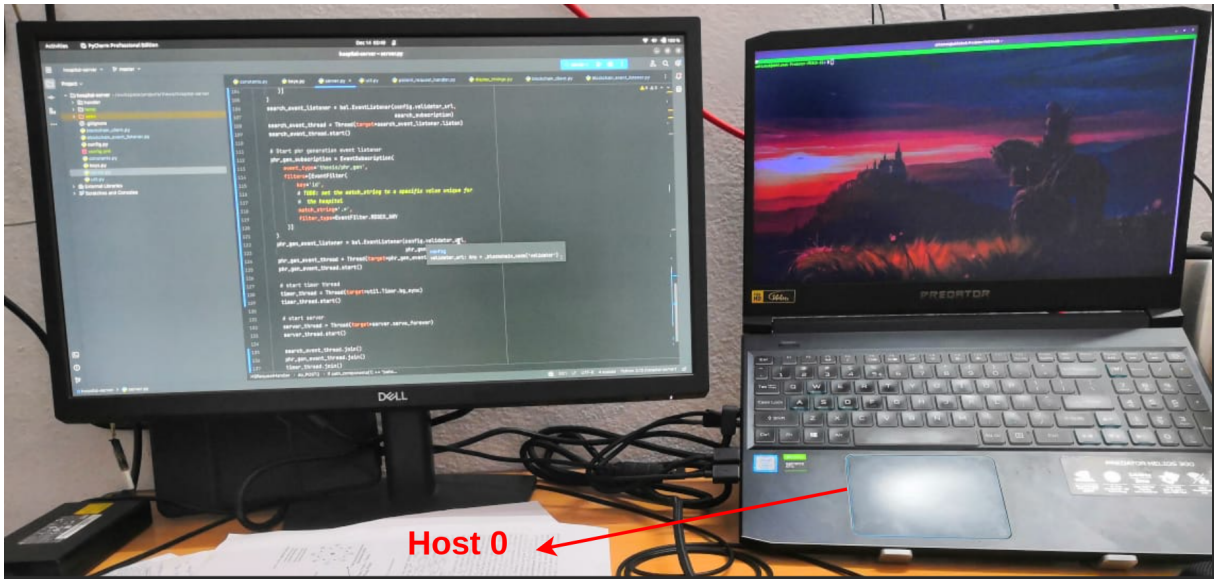

Figure 4.8: Experimental setup 2

PHR Addition

```
------------------------------------------------------------
1. Register
2. Request PHR
3. Search
4. Exit
Choice: 2
User Name: yen
Password: yen
Doctor ID: doc1
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:9000
Starting new HTTP connection (1): localhost:9000
DEBUG:urllib3.connectionpool:http://localhost:9000 "POST /patient/phr_gen HTTP/1.1" 200 None
http://localhost:9000 "POST /patient/phr_gen HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:9000
Starting new HTTP connection (1): localhost:9000
DEBUG:urllib3.connectionpool:http://localhost:9000 "POST /patient/entries HTTP/1.1" 200 None
http://localhost:9000 "POST /patient/entries HTTP/1.1" 200 None
DEBUG:root:entries submitted successfully
entries submitted successfully
PHR Generated Successfully
<shelve.DbfilenameShelf object at 0x7f7d263f1720>
============================================================
```

Figure 4.9: Logs output - PHR addition

Keyword Search

```
============================================================
1. Register
2. Request PHR
3. Search
4. Exit
Choice: 3
User Name: yen
Keyword: trinity
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:9000
Starting new HTTP connection (1): localhost:9000
DEBUG:urllib3.connectionpool:http://localhost:9000 "POST /external/search HTTP/1.1" 200 None
http://localhost:9000 "POST /external/search HTTP/1.1" 200 None
DEBUG:root:search request successfull
search request successfull
b'QmUxPnqqXUzHvuUkKSZYF38PwUhv3DpZ7wEckCANNZ86vK'
fid list:  [b'QmUxPnqqXUzHvuUkKSZYF38PwUhv3DpZ7wEckCANNZ86vK']
<shelve.DbfilenameShelf object at 0x7f7d263f1720>
============================================================
```

Figure 4.10: Logs output - Keyword search

4.4 Performance evaluation

In this section, we present the results of evaluation of the proposed scheme under different conditions, and also provide a comparative study with the existing competing schemes.

4.5 Theoretical analysis

We provide an asymptotic analysis of various algorithms presented in our scheme.

- *Keyword extraction:* This algorithm iterates over all the keywords present in the PHR. Therefore, it has time complexity of $O(|W|)$, where, $|W|$ is the number of keywords present in the PHR.
- *Secure entries generation:* The most time consuming operation in this algorithm is the for-loop which iterates over the set of keywords V. Therefore, the complexity is $O(|V|)$.
- *Trapdoor generation*: All the operations present in this algorithm have constant execution time. Therefore, the time complexity is $O(1)$.

Table 4.2: Communication cost

Phase		Cost (in bytes)
Add	PHR request	$32 + m$
	PHR upload	$m + 32$
	Secure entries upload	$80\|\Delta\|$
Search	Trapdoor request	$\|w\| + 33 + 48$
	Search request	$48 + (32 + 80)\|\Gamma\| + 32\|\Gamma\|$
	Identifier retrieval	$2 \times 32\|\Gamma\|$
	PHR retrieval	$32 + m$

- *Keyword search:* The most time consuming operation here is the while-loop which runs till the complete list of nodes is extracted. Hence, the time complexity is $O(|\Gamma|)$ where Γ is the set identifiers corresponding to a keyword in the encrypted index.

- *Retrieve identifiers:* In this algorithm, the most time consuming operation is the for-loop which iterates over each element of Γ. Hence, the time complexity is $O(|\Gamma|)$.

In Table 4.2, we show the total communication overhead of our scheme. Here, 32 is the size of a key of the index Δ, or the size of the PHR identifier, 33 is the size of public key, 48 is the size of trapdoor, 80 is the size of each entry in Δ or the size of contents of a node in encrypted index and m is the size of PHR.

4.6 Test-bed setup and organization

Figure 4.11 shows the organization of our test-bed setup. We used four desktop systems shown in Figure 4.7 and a laptop shown in Figure 4.8 as the computing hardware for our simulation. Each of the desktops had an Intel core i5 12400 processor, 16 GB RAM, and 1TB SSD, and the laptop had an Intel core i7-9750H processor, 16 GB RAM, and 256 GB SSD. Each of the desktops ran Ubuntu Server 22.04 LTS and had Docker (version 20.10.22) installed on it. Docker was used to simulate the Hyperledger Sawtooth nodes and IPFS nodes. The laptop used Ubuntu Desktop 22.04 LTS as the Operating System (OS). It ran an instance of the hospital server, doctor device, and patient device. All the

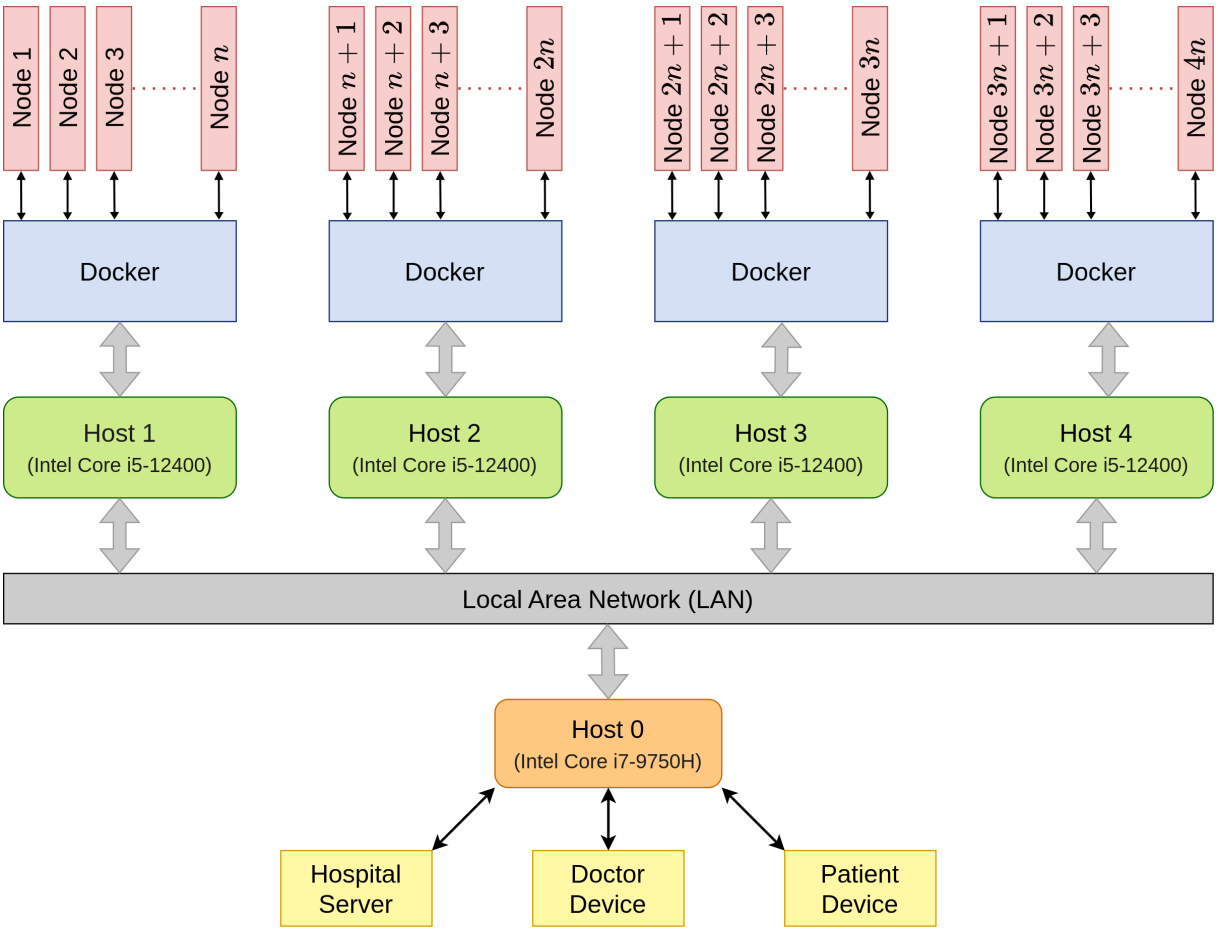

Figure 4.11: Experimental setup organization

systems were part of the same LAN. The entire test-bed setup took around 4000 lines of code written mainly in Python (version 3.10.6). We used the *cryptography* Python library for various cryptographic operations. Figures 4.9 and 4.10 show the outputs logs during execution of our scheme. The entire code-base and instructions on how to run it are available at the following url - https://github.com/abhishekbisht1429/phr-ss-system

4.6.1 Experimental results

We used the Secure Hash Algorithm (SHA-256) as the replacement of the hash function h. For ECIES, we use the combination of ECDH key exchange and Advanced Encryption Standard (AES), wherein ECDH uses the standard SECP256-R1 curve specified by the "National Institute of Standards and Technology (NIST)" and AES uses offset codebook

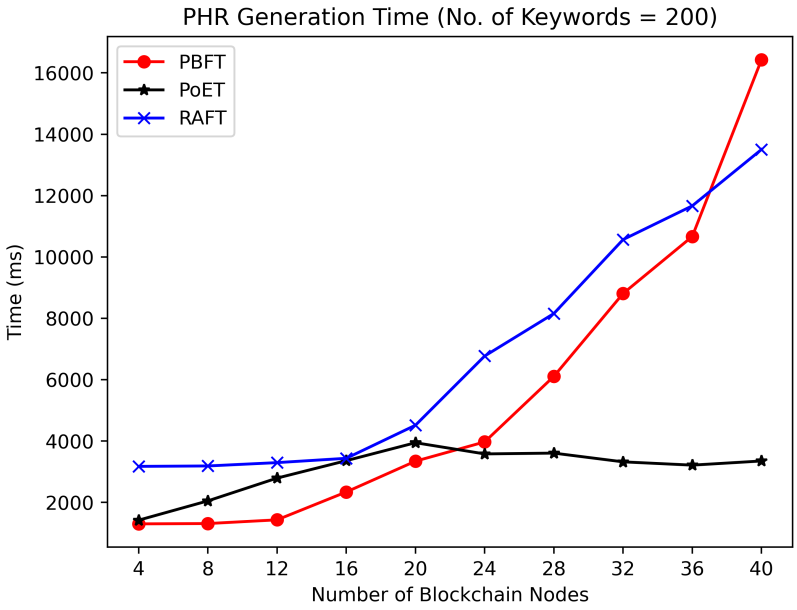

Figure 4.12: Experiment 1: Results

version 3 (OCB3) mode. For symmetric key encryption and decryption, AES in OCB3 mode is used. Table 4.3 shows the timings of the core operations that have been used. The timings shown in this table were calculated by taking the average over timings of 1000 executions for an input size of 256 bytes, and the hardware used was Intel Core i7-9750H @ 2.60 Hz, 16 GB RAM, and 256 GB solid-state drive (SSD).

Next, we present the extensive simulation results for our scheme by presenting the results of six experiments that we carried out after the implementation of the proposed scheme. For the experiments, we have taken a minimum number of unique keywords in a PHR to be 200. This value is based on the Heaps' law [126] for English text (with $K = 3.67$ and $\beta = 0.69$ [5]), and on the fact that a standard A4 page on an average contains 350 – 450 words. Moreover, due to limitations of computational hardware available to us, the maximum number of blockchain nodes that we were able to use for the experiments was 40. Beyond that number, other factors such as context switching would influence the actual results to a large extent.

[5]Note: β here is not related to the scheme, but is a part of the formula in Heaps' law.

Table 4.3: Timings of core operations

Operation	Time (ms)
SHA-256	0.0014
AES-OCB3 Encryption	0.0161
AES-OCB3 Decryption	0.0149
ECIES Encryption (ECDH + AES-OCB3)	0.3425
ECIES Decryption (ECDH + AES-OCB3)	0.6801

1) Experiment 1

For this, we used a PHR with 200 unique keywords and recorded the overall time taken to add it to the system by varying the number of nodes in the blockchain network. We recorded the observations for three different state-of-the-art consensus algorithms: 1) PBFT, 2) PoET, and 3) Raft, and presented the results in Figure 4.12. For the number of nodes less than 20, PBFT performs better than the other two algorithms. On further increasing the number of nodes, the performance of PBFT and Raft declines exponentially. However, for a larger number of nodes, PoET works quite well.

2) Experiment 2

We added a PHR containing 200 unique keywords to the system for this. We then searched for a keyword and recorded the time taken by the search algorithm against the number of blockchain nodes. Figure 4.13 shows the results using PBFT, PoET, and Raft. It can be seen that the increase in the number of nodes does not affect the search time. This is because our search algorithm directly queries the state of the blockchain using REST API and thus does not has to be processed as a transaction by the blockchain nodes.

3) Experiment 3

In this experiment, we fixed the number of blockchain nodes to 8 and then recorded the time taken to complete PHR addition against the number of keywords in the PHR. Figure 4.14 shows the results using PBFT, PoET, and Raft. The results show that the addition time increases linearly with the number of keywords. However, PBFT performs

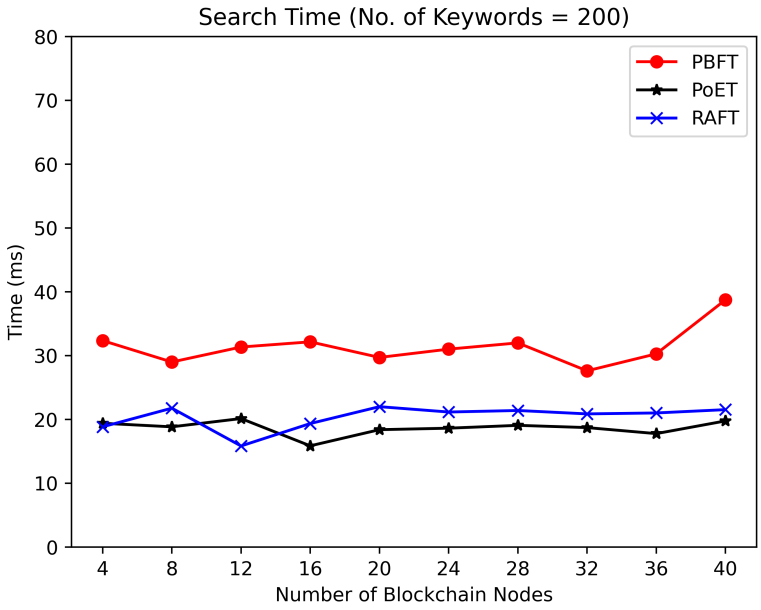

Figure 4.13: Experiment 2: Results

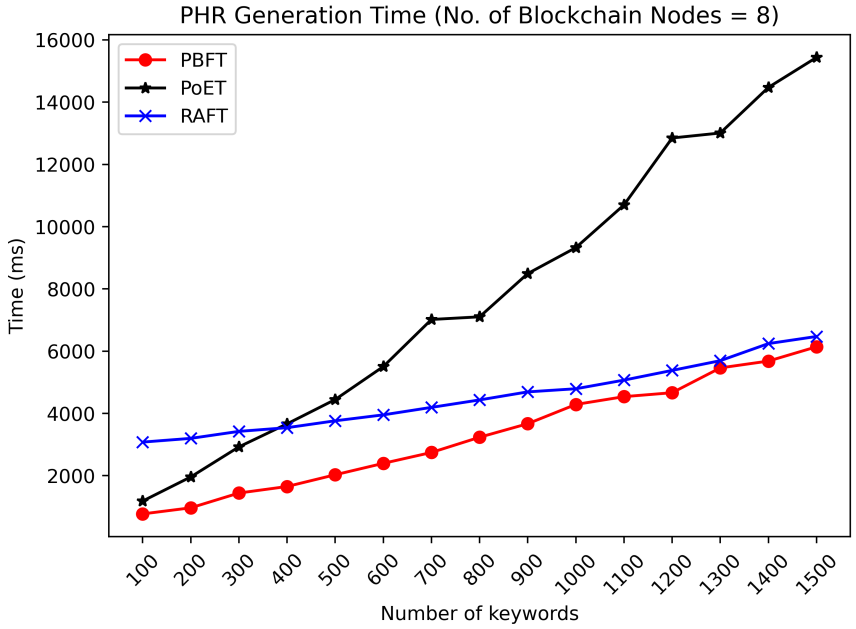

Figure 4.14: Experiment 3: Results

better than the other two algorithms.

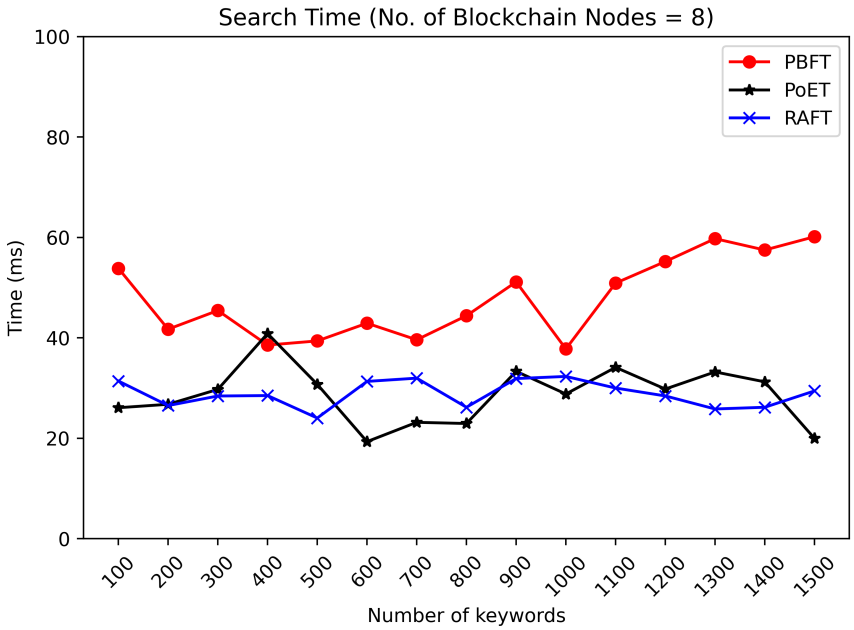

Figure 4.15: Experiment 4: Results

4) Experiment 4

We fixed the number of blockchain nodes to 8, added a PHR, and recorded the time to search a keyword against the number of unique keywords in the PHR. For this experiment, we have taken a list of size one by adding only one PHR to obtain consistent results. The reason is that the asymptotic complexity of the search is linear in terms of entries in the linked list corresponding to the keyword being searched. So, having a linked list of different sizes will affect the search results, eventually leading to inconsistent figures. Figure 4.15 shows the results obtained using PBFT, PoET, and Raft, and it can be concluded that search time is not affected by the number of keywords present in the PHR. This is because the MR-tree used by sawtooth has a constant lookup time.

5) Experiment 5

In this experiment, we measured the effect of change in the number of IPFS nodes on PHR upload time. Figure 4.16 shows that it remains unaffected by the change.

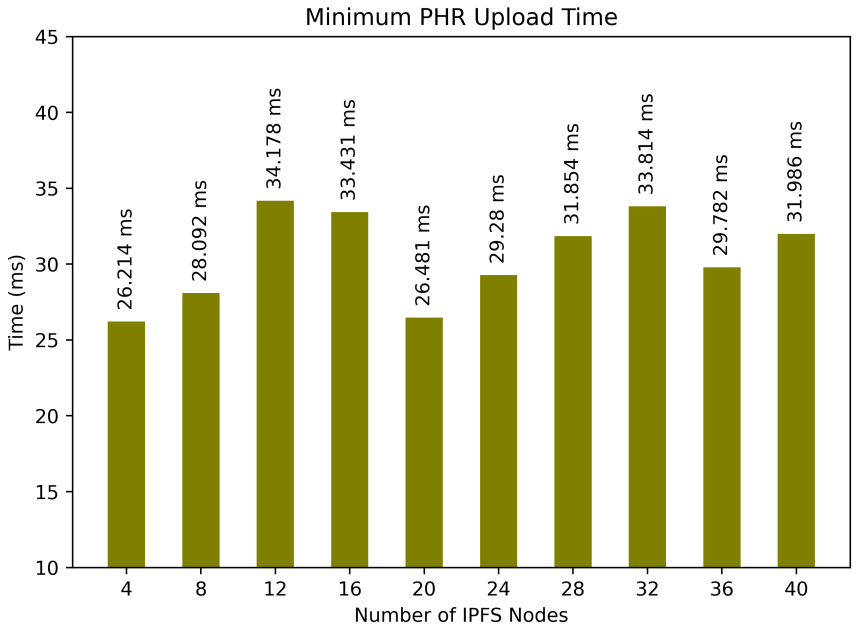

Figure 4.16: Experiment 5: Results

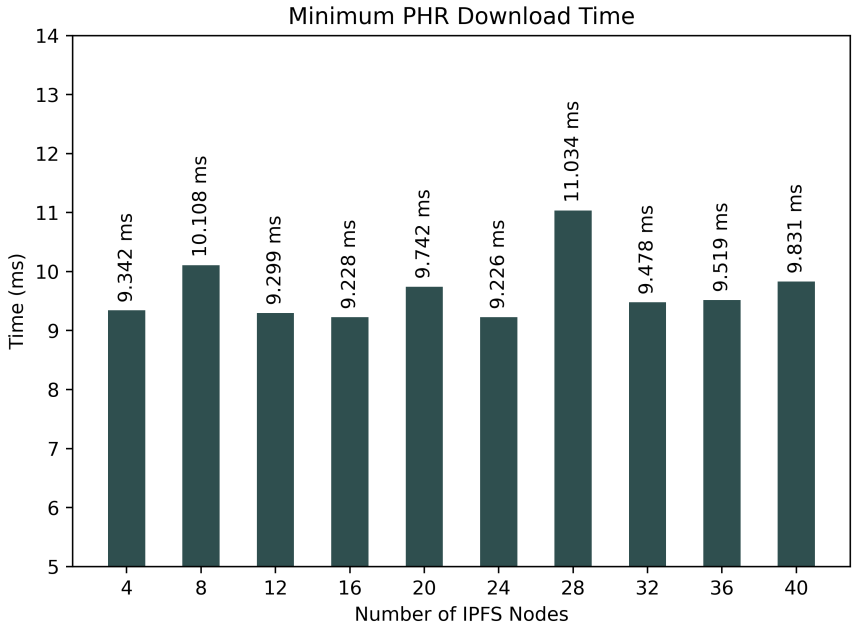

Figure 4.17: Experiment 6: Results

6) Experiment 6

Here, we measured the effect of the change in the number of IPFS nodes on PHR download time. Figure 4.17 shows that it remains unaffected by the change.

4.6.2 Comparative analysis

Although there is a plethora of work on SSE using blockchain, there are only few works on its application in PHR sharing. However, many PHR-sharing schemes are based on public key-based searchable encryption. A strict comparison is not possible due to the different cryptographic parameters used by these schemes. Nevertheless, in Table 4.4, we have tried to compare these schemes with our scheme based on the expected features from a PHR sharing scheme. An exception to the above is a scheme proposed by Tang *et al.*[65], which is based on symmetric searchable encryption and uses a technique similar to our scheme for adding PHRs to the system. However, it relies on the local storage of hospitals, unlike our scheme, which uses the decentralized storage, and thus, it has limited practical applications. Our scheme, though primarily based on symmetric searchable encryption, uses elliptic curve cryptography for communication between entities and, therefore, can be categorized as a hybrid scheme.

Table 4.4: Comparison of security and functional features

Scheme	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
Wang *et al.* [61]	Asymmetric	✓	✓	×	✓	✓	✓	×
Sun *et al.* [113]	Asymmetric	✓	✓	×	✓	✓	✓	×
Zhang *et al.* [6]	Asymmetric	✓	×	×	×	✓	✓	×
Tang *et al.* [65]	Symmetric	✓	✓	✓	✓	✓	×	✓
Proposed	Symmetric	✓	✓	✓	✓	✓	✓	✓

Note: A_1: Encryption Type, A_2: Confidentiality, A_3: Dynamic, A_4: Forward Secure, A_5: Verifiable, A_6: Blockchain Based, A_7: Decentralized Storage, A_8: Efficient

In Table 4.5, we show a comparative analysis on the time complexities of various operations involved in the proposed scheme and other existing schemes. For the setup phase, the time complexity for each scheme is constant. However, the asymmetric schemes will be comparatively slower because of the use of bilinear pairings.

All other schemes than the scheme proposed in [6], which generates index only once at the beginning, can dynamically add PHRs as and when required. W represents the set of unique keywords in a PHR and D is the set of PHRs used during the build phase. A

Table 4.5: Time complexity comparison

Scheme	**Setup**	**Build**	**Add**	**Search**
Wang *et al.* [61]	$O(1)$	–	$O(\|W\| + \|A\|)$	$O(\|W\| + \|A\|)$
Sun *et al.* [113]	$O(1)$	–	$O(\|W\|)$	$O(\|A\|$
Zhang *et al.* [6]	$O(1)$	$O(\|D\| \cdot \|W\|)$	–	$O(\|D\|)$
Tang *et al.* [65]	$O(1)$	–	$O(\|W\|$	$O(\|\Gamma\|)$
Proposed	$O(1)$	–	$O(\|V\|)$	$O(\|\Gamma\|)$

presents in complexities of some of the schemes is related to attribute based encryption, where it denotes the set of attributes. In case of our scheme, V denotes the super-set of keyword set W and Γ is the set of file identifiers corresponding to a single keyword in the index. For most of the patients, the PHRs will not be very large in number, and therefore, comparatively our scheme is quite efficient during search.

4.7 Summary

With the growing dependency of healthcare on IoMT infrastructure, we require a secure system that not only provides confidentiality and privacy to the patients' health data, but also allows its secure sharing with other parties in the healthcare ecosystem. We attempted to provide a solution by proposing a novel scheme for PHR sharing that is dynamic, efficient and practically implementable. Our design considered various entities into consideration, including IoMT smart devices, and provided a carefully designed protocol for their interactions. We proved the proposed scheme to be forward secure, verifiable and semantically secure against an adaptive adversary. Moreover, we based our scheme on top of a decentralized file system (IPFS) in order to make our scheme fault tolerant and robust against attacks targeting centralized systems. Additionally, we provided various insights into how this scheme can be extended to support parallelization and conjunctive multi-keyword search.

Chapter 5

Conclusion and Open Research Challenges

5.1 Research contributions

In this thesis, we discussed about the importance of security and privacy of health data and discussed current technologies that are in use for their storage and sharing. We provided a detailed survey of existing works on sharing and storage of health data using SSE and blockchain. We discussed the shortcomings of these schemes in terms of security and efficiency and then proposed a novel scheme with better security and better efficiency using SSE, blockchain and IPFS. We have provided formal security proofs for our scheme that prove it to be forward secure, verifiable and semantically secure against an adaptive adversary. The use of decentralized file system (IPFS) makes our scheme secure against attacks targeting centralized systems. Moreover, we have provided test-bed experiment results that demonstrate practicality of our proposed solution.

5.2 Open research problems

Searchable encryption with blockchain has opened up avenues for construction of health care systems that not only allow secure storage, but also secure sharing of health records under the control of the data owner. The goal of research in this domain should be to construct a universal healthcare infrastructure that is easily adaptable and is convenient

to use for the users apart from being secure. However, there are still many open challenges that need to be tackled before we reach to this goal. We discuss some of them as follows:

1. There is a need for a universal standard for the format of personal health records (PHRs). It will be easier to share PHRs between different medical institutions and other third parties involved.

2. We emphasis on the need for SE schemes that will allow more expressive queries rather than a single keyword search and still maintain the security of the PHRs.

3. There is also need for more SE schemes that should leverage the advantage of parallel computation as well.

5.3 Future works

In this section, we list the following future research directions that we would like to work in our future study.

5.3.1 Support for parallelization

The addition of PHR in our scheme supports parallelization. As seen in"Uploading to blockchain" part of Section 4.2.3, the addition of $(addr, val)$ pairs to the state of the blockchain are independent of each other. Therefore, it can be modified to leverage parallelization, significantly decreasing PHR addition time. However, the algorithm must run as a part of a transaction processor. As a result, all the blockchain peers must have parallelization capability; otherwise, a single peer without parallelization would create a bottleneck for the entire system.

5.3.2 Conjunctive multi-keyword search support

Our scheme can be modified to support multi-keyword search, but at the expense of more leakage. The modified scheme would require the PHR ids to be encrypted using a deterministic algorithm. The search algorithm executed by the hospital server can take a set of keywords as input and obtain the linked lists corresponding to the keywords. Subsequently, it can find the common encrypted PHR ids and return them to the user.

Moreover, the process to obtain the linked lists can be made parallel because each can be obtained independently of the other.

Bibliography

[1] B. Bera, "Design and Analysis of Blockchain-Based Access Control Protocols for Internet of Drones," Ph.D. dissertation, International Institute of Information Technology, Hyderabad, India, Center for Security, Theory and Algorithmic Research, November 2022.

[2] "Hyperledger Sawtooth Architecture Guide, Intel Corporation," 2020, https://sawtooth.hyperledger.org/docs/1.2/. Accessed on June 2023.

[3] "Global State — sawtooth.hyperledger.org," https://sawtooth.hyperledger.org/docs/1.2/architecture/global_state.html, accessed on April 2023.

[4] A. Vangala, A. K. Das, N. Kumar, and M. Alazab, "Smart Secure Sensing for IoT-Based Agriculture: Blockchain Perspective," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 591–17 607, 2021.

[5] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 131–144, 2018.

[6] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, vol. 6, pp. 31 077–31 087, 2018.

[7] H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," *Computers & Electrical Engineering*, vol. 73, pp. 32–45, 2019.

[8] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an Encrypted Cloud Meets Blockchain: A Decentralized, Reliable and Fair Realization," in *IEEE Conference on Computer Communications (INFOCOM'18)*, Honolulu, HI, USA, 2018, pp. 792–800.

[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A berkeley view of cloud computing," Technical Report UCB/EECS-2009-28, EECS Department, University of California ..., Tech. Rep., 2009.

[10] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," 2011-09-28 2011.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Cryptography Mailing list at https://metzdowd.com*, March 2009.

[12] B. Bera, A. Mitra, A. K. Das, D. Puthal, and Y. Park, "Private Blockchain-Based AI-Envisioned Home Monitoring Framework in IoMT-Enabled COVID-19 Environment," *IEEE Consumer Electronics Magazine*, vol. 12, no. 3, pp. 62–71, 2023.

[13] M. Wazid, A. K. Das, and Y. Park, "Blockchain-enabled secure communication mechanism for IoT-driven personal health records," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 4, p. e4421, 2022.

[14] B. Bera, A. K. Das, and S. K. Das, "Search on Encrypted COVID-19 Healthcare Data in Blockchain-Assisted Distributed Cloud Storage," *IEEE Internet of Things Magazine*, vol. 4, no. 4, pp. 127–132, 2021.

[15] A. K. Das, B. Bera, and D. Giri, "AI and Blockchain-Based Cloud-Assisted Secure Vaccine Distribution and Tracking in IoMT-Enabled COVID-19 Environment," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 26–32, 2021.

[16] S. Son, J. Lee, M. Kim, S. Yu, A. K. Das, and Y. Park, "Design of Secure Authentication Protocol for Cloud-Assisted Telecare Medical Information System Using Blockchain," *IEEE Access*, vol. 8, pp. 192 177–192 191, 2020.

[17] M. Wazid, B. Bera, A. Mitra, A. K. Das, and R. Ali, "Private Blockchain-Envisioned Security Framework for AI-Enabled IoT-Based Drone-Aided Healthcare Services," in *2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond (DroneCom'20)*, London, United Kingdom, 2020, pp. 37–42.

[18] M. Wazid, A. K. Das, R. Hussain, N. Kumar, and S. Roy, "BUAKA-CS: Blockchain-enabled user authentication and key agreement scheme for crowdsourcing system," *Journal of Systems Architecture*, vol. 123, p. 102370, 2022.

[19] A. K. Das, B. Bera, S. Saha, N. Kumar, I. You, and H.-C. Chao, "AI-Envisioned Blockchain-Enabled Signature-Based Key Management Scheme for Industrial Cyber-Physical Systems," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6374–6388, 2022.

[20] M. Wazid, B. Bera, A. K. Das, S. Garg, D. Niyato, and M. S. Hossain, "Secure Communication Framework for Blockchain-Based Internet of Drones-Enabled Aerial Computing Deployment," *IEEE Internet of Things Magazine*, vol. 4, no. 3, pp. 120–126, 2021.

[21] B. Bera, M. Wazid, A. K. Das, and J. J. P. C. Rodrigues, "Securing Internet of Drones Networks Using AI-Envisioned Smart-Contract-Based Blockchain," *IEEE Internet of Things Magazine*, vol. 4, no. 4, pp. 68–73, 2021.

[22] B. Bera, A. K. Das, and A. K. Sutrala, "Private blockchain-based access control mechanism for unauthorized UAV detection and mitigation in Internet of Drones environment," *Computer Communications*, vol. 166, pp. 91–109, 2021.

[23] B. Bera, D. Chattaraj, and A. K. Das, "Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment," *Computer Communications*, vol. 153, pp. 229–249, 2020.

[24] A. Mitra, B. Bera, and A. K. Das, "Design and Testbed Experiments of Public Blockchain-Based Security Framework for IoT-Enabled Drone-Assisted Wildlife Monitoring," ser. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2021, pp. 1–6.

[25] B. Bera, A. Vangala, A. K. Das, P. Lorenz, and M. K. Khan, "Private blockchain-envisioned drones-assisted authentication scheme in IoT-enabled agricultural environment," *Computer Standards & Interfaces*, vol. 80, p. 103567, 2022.

[26] A. Vangala, A. K. Sutrala, A. K. Das, and M. Jo, "Smart Contract-Based Blockchain-Envisioned Authentication Scheme for Smart Farming," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 792–10 806, 2021.

[27] D. Chattaraj, B. Bera, A. K. Das, S. Saha, P. Lorenz, and Y. Park, "Block-CLAP: Blockchain-Assisted Certificateless Key Agreement Protocol for Internet of Vehicles

in Smart Transportation," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8092–8107, 2021.

[28] A. Vangala, B. Bera, S. Saha, A. K. Das, N. Kumar, and Y. Park, "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in Intelligent Transportation Systems," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15 824–15 838, 2021.

[29] P. Bagga, A. K. Sutrala, A. K. Das, and P. Vijayakumar, "Blockchain-based batch authentication protocol for Internet of Vehicles," *Journal of Systems Architecture*, vol. 113, p. 101877, 2021.

[30] B. Bera, S. Saha, A. K. Das, N. Kumar, P. Lorenz, and M. Alazab, "Blockchain-Envisioned Secure Data Delivery and Collection Scheme for 5G-Based IoT-Enabled Internet of Drones Environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9097–9111, 2020.

[31] D. Chattaraj, B. Bera, A. K. Das, J. J. P. C. Rodrigues, and Y. Park, "Designing Fine-Grained Access Control for Software-Defined Networks Using Private Blockchain," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1542–1559, 2022.

[32] R. Shashidhara, N. Ahuja, M. Lajuvanthi, S. Akhila, A. K. Das, and J. J. P. C. Rodrigues, "SDN-chain: Privacy-preserving protocol for software defined networks using blockchain," *Security and Privacy*, vol. 4, no. 6, p. e178, 2021.

[33] S. Banerjee, B. Bera, A. K. Das, S. Chattopadhyay, M. K. Khan, and J. J. Rodrigues, "Private blockchain-envisioned multi-authority CP-ABE-based user access control scheme in IIoT," *Computer Communications*, vol. 169, pp. 99–113, 2021.

[34] S. Saha, D. Chattaraj, B. Bera, and A. Kumar Das, "Consortium blockchain-enabled access control mechanism in edge computing based generic Internet of Things environment," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 6, p. e3995, 2021.

[35] B. Bera, A. K. Das, M. S. Obaidat, P. Vijayakumar, K.-F. Hsiao, and Y. Park, "AI-Enabled Blockchain-Based Access Control for Malicious Attacks Detection and Mitigation in IoE," *IEEE Consumer Electronics Magazine*, vol. 10, no. 5, pp. 82–92, 2021.

[36] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5744–5761, 2021.

[37] S. Jangirala, A. K. Das, and A. V. Vasilakos, "Designing Secure Lightweight Blockchain-Enabled RFID-Based Authentication Protocol for Supply Chains in 5G Mobile Edge Computing Environment," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7081–7093, 2020.

[38] Y. Qian, Y. Jiang, L. Hu, M. S. Hossain, M. Alrashoud, and M. Al-Hammadi, "Blockchain-Based Privacy-Aware Content Caching in Cognitive Internet of Vehicles," *IEEE Network*, vol. 34, no. 2, pp. 46–51, 2020.

[39] A. M. Saghiri, M. Vahdati, K. Gholizadeh, M. R. Meybodi, M. Dehghan, and H. Rashidi, "A framework for cognitive Internet of Things based on blockchain," ser. 4th International Conference on Web Research (ICWR'18), Tehran, Iran, 2018, pp. 138–143.

[40] M. Vahdati, K. Gholizadeh HamlAbadi, A. M. Saghiri, and H. Rashidi, "A Self-Organized Framework for Insurance Based on Internet of Things and Blockchain," ser. IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud'18), Barcelona, Spain, 2018, pp. 169–175.

[41] M. Jakobsson and A. Juels, *Proofs of Work and Bread Pudding Protocols*, ser. Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium, Boston, MA, 1999, pp. 258–272.

[42] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.

[43] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," ser. USENIX Annual Technical Conference (Usenix ATC 14), Philadelphia, PA, USA, 2014, pp. 305–319.

[44] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, 1980.

[45] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[46] F. Saleh, "Blockchain without waste: Proof-of-stake," *The Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, 2021.

[47] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism amp; Consortium Blockchain," ser. IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Bangkok, Thailand, 2017, pp. 466–473.

[48] G. Wood *et al.*, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[49] I. Bentov, R. Pass, and E. Shi, "Snow White: Provably Secure Proofs of Stake," *IACR Cryptology ePrint Archive*, vol. 2016, no. 919, 2016.

[50] S. King and S. Nadal, "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:42319203

[51] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," ser. Advances in Cryptology (CRYPTO'17), Santa Barbara, CA, USA, 2017, pp. 357–388.

[52] J. Kwon, "Tendermint: Consensus without mining," *Self-Published Paper (Draft v.0.6)*, vol. 1, no. 11, 2014, accessed on April 2023. [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf

[53] "The XRP Ledger," 2014, accessed on March 2023. [Online]. Available: https://xrpl.org/consensus-principles-and-rules.html

[54] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain," *Journal of Medical Systems*, vol. 42, no. 8, pp. 1–18, 2018.

[55] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, C. Yuen, S. Sun, K.-Y. Lam, and L. H. Koh, "Blockchain for the Internet of Vehicles Towards Intelligent Transportation Systems: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4157–4185, 2021.

[56] V. B. et al., "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, no. 37, pp. 2–1, 2014.

[57] J. S. Sara Saberi, Mahtab Kouhizadeh and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019. [Online]. Available: https://doi.org/10.1080/00207543.2018.1533261

[58] J. Golosova and A. Romanovs, "The Advantages and Disadvantages of the Blockchain Technology," ser. 6th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE). Vilnius, Lithuania, 2018, pp. 1–6.

[59] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy (S&P'00)*, Berkeley, California, USA, 2000, pp. 44–55.

[60] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *arXiv*, 2014, accessed on February 2023. [Online]. Available: https://arxiv.org/abs/1407.3561

[61] Y. Wang, A. Zhang, P. Zhang, Y. Qu, and S. Yu, "Security-Aware and Privacy-Preserving Personal Health Record Sharing Using Consortium Blockchain," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 014–12 028, 2022.

[62] Y. Wang, A. Zhang, P. Zhang, and H. Wang, "Cloud-assisted EHR sharing with security and privacy preservation via consortium blockchain," *IEEE Access*, vol. 7, pp. 136 704–136 719, 2019.

[63] M. Alsayegh, T. Moulahi, A. Alabdulatif, and P. Lorenz, "Towards Secure Searchable Electronic Health Records Using Consortium Blockchain," *Network*, vol. 2, no. 2, pp. 239–256, 2022.

[64] L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Generation Computer Systems*, vol. 95, pp. 420–429, 2019.

[65] X. Tang, C. Guo, K.-K. R. Choo, Y. Liu, and L. Li, "A secure and trustworthy medical record sharing scheme based on searchable encryption and blockchain," *Computer Networks*, vol. 200, p. 108540, 2021.

[66] V. Shoup, "A Proposal for an ISO Standard for Public Key Encryption," Cryptology ePrint Archive, Paper 2001/112, 2001, accessed on November 2022. [Online]. Available: https://eprint.iacr.org/2001/112

[67] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. CRC press, 2020.

[68] D. Boneh, "Pairing-based cryptography: Past, present, and future," in *Advances in Cryptology – ASIACRYPT 2012*, X. Wang and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 1.

[69] A. Menezes, "An Introduction to Pairing-Based Cryptography," 2013, https://www.math.uwaterloo.ca/~ajmeneze/publications. Accessed on May 2020.

[70] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," vol. 21, no. 2, p. 120–126, feb 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[71] W. E. May, "Secure Hash Standard," 2015, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. Accessed on May 2022.

[72] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[73] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," ser. 2016 IEEE symposium on security and privacy (SP). San Jose, CA, USA: IEEE, 2016, pp. 839–858.

[74] "Sawtooth — hyperledger.org," https://www.hyperledger.org/projects/sawtooth, accessed on April 2023.

[75] T. Krovetz and P. Rogaway, "The OCB Authenticated-Encryption Algorithm," RFC 7253, May 2014, accessed on November 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc7253

[76] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A Survey of Provably Secure Searchable Encryption," *ACM Computing Surveys*, vol. 47, no. 2, August 2014.

[77] Y. Wang, J. Wang, and X. Chen, "Secure searchable encryption: a survey," *Journal of Communications and Information Networks*, vol. 1, pp. 52–65, 2016.

[78] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable Symmetric Encryption: Designs and Challenges," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–37, 2017, Article No. 40.

[79] R. Zhang, R. Xue, and L. Liu, "Searchable Encryption for Healthcare Clouds: A Survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 978–996, 2018.

[80] H. Pham, J. Woodworth, and M. Amini Salehi, "Survey on secure search over encrypted data on the cloud," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 17, p. e5284, 2019.

[81] N. Andola, R. Gahlot, V. K. Yadav, S. Venkatesan, and S. Verma, "Searchable encryption on the cloud: a survey," *The Journal of Supercomputing*, vol. 78, no. 7, pp. 9952–9984, 2022.

[82] O. Goldreich and R. Ostrovsky, "Software Protection and Simulation on Oblivious RAMs," *Journal of the ACM*, vol. 43, no. 3, pp. 431–473, May 1996.

[83] E.-J. Goh, "Secure Indexes," Cryptology ePrint Archive, Paper 2003/216, 2003, accessed on May 2022. [Online]. Available: https://eprint.iacr.org/2003/216

[84] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[85] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," ser. International Conference on the Theory and Applications of Cryptographic Techniques – Advances in Cryptology (EUROCRYPT'04), C. Cachin and J. L. Camenisch, Eds., Interlaken, Switzerland, 2004, pp. 506–522.

[86] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," ser. 21st Annual International Cryptology Conference – Advances in Cryptology (CRYPTO'01), J. Kilian, Ed., Santa Barbara, California, USA, 2001, pp. 213–229.

[87] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," ser. International Conference on Applied Cryptography and Network Security (ACNS'05), New York, NY, USA, 2005, pp. 442–455.

[88] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," ser. ACM conference on Computer and Communications Security (CCS'12), Raleigh, North Carolina, USA, 2012, pp. 965–976.

[89] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation," in *Network and Distributed System Security Symposium*. Citeseer, 2012, pp. 1–12, accessed on January 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:12703012

[90] E. Stefanov, C. Papamanthou, and E. Shi, "Practical Dynamic Searchable Encryption with Small Leakage," Cryptology ePrint Archive, Paper 2013/832, 2013, https://eprint.iacr.org/2013/832. Accessed on March 2022.

[91] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," ser. International Conference on Financial Cryptography and Data Security. Okinawa, Japan: Springer, 2013, pp. 258–274.

[92] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual Cryptology Conference (CRYPTO'13)*, Santa Barbara, California, USA, 2013, pp. 353–373.

[93] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," ser. 22nd ACM Conference on Computer and Communications Security (CCS), Denver, Colorado, USA, 2015, pp. 668–679.

[94] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable Dynamic Symmetric Searchable Encryption: Optimality and Forward Security," Cryptology ePrint Archive, Paper 2016/062, 2016, accessed on May 2023. [Online]. Available: https://eprint.iacr.org/2016/062

[95] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," ser. Proceedings of the

2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, USA, 2017, pp. 1465–1482.

[96] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," ser. ACM Conference on Computer and Communications Security (CCS), Dallas, TX, USA, 2017, pp. 1449–1463.

[97] M. Etemad, A. Küpçü, C. Papamanthou, and D. Evans, "Efficient Dynamic Searchable Encryption with Forward Privacy," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, pp. 20 – 5, 2017, accessed on May 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:3836548

[98] Y. Watanabe, K. Ohara, M. Iwamoto, and K. Ohta, "Efficient Dynamic Searchable Encryption with Forward Privacy under the Decent Leakage," ser. Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy, Baltimore, MD, USA, 2022, pp. 312–323.

[99] G. Asharov, G. Segev, and I. Shahaf, "Tight tradeoffs in searchable symmetric encryption," *Journal of Cryptology*, vol. 34, no. 2, pp. 1–37, 2021.

[100] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security," ser. European Symposium on Research in Computer Security (ESORICS'18). Barcelona, Spain: Springer, 2018, pp. 228–246.

[101] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 912–927, 2018.

[102] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," ser. ACM Conference on Computer and Communications Security (CCS), Toronto, Canada, 2018, pp. 1038–1055.

[103] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou, "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 460–474, 2019.

[104] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption with forward and stronger backward privacy," ser. European Symposium on Research in Computer Security (ESORICS'19). Luxembourg: Springer, 2019, pp. 283–303.

[105] L. Chen, J. Li, and J. Li, "Toward Forward and Backward Private Dynamic Searchable Symmetric Encryption Supporting Data Deduplication and Conjunctive Queries," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 17 408–17 423, 2023.

[106] Y. Guo, C. Zhang, and X. Jia, "Verifiable and Forward-secure Encrypted Search Using Blockchain Techniques," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[107] R. Du and Y. Wang, "Verifiable Blockchain-Based Searchable Encryption with forward and backward privacy," ser. 16th International Conference on Mobility, Sensing and Networking (MSN). Tokyo, Japan: IEEE, 2020, pp. 630–635.

[108] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," ser. 2016 2nd international conference on open and big data (OBD). Vienna, Austria: IEEE, 2016, pp. 25–30.

[109] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of Medical Systems*, vol. 40, no. 10, pp. 1–8, 2016.

[110] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, p. 44, 2017.

[111] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of Medical Systems*, vol. 42, no. 8, pp. 1–11, 2018.

[112] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," *International Journal of Information Security*, vol. 14, no. 6, pp. 487–497, Nov 2015.

[113] J. Sun, L. Ren, S. Wang, and X. Yao, "A blockchain-based framework for electronic medical records sharing with fine-grained access control," *PLOS ONE*, vol. 15, no. 10, p. e0239946, 2020.

[114] S. Niu, L. Chen, J. Wang, and F. Yu, "Electronic health record sharing scheme with searchable attribute-based encryption on blockchain," *IEEE Access*, vol. 8, pp. 7195–7204, 2019.

[115] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102 887–102 901, 2019.

[116] X. Nie, A. Zhang, J. Chen, Y. Qu, and S. Yu, "Time-Enabled and Verifiable Secure Search for Blockchain-Empowered Electronic Health Record Sharing in IoT," *Security and Communication Networks*, vol. 2022, p. 1103863, Dec 2022.

[117] J. Benet, "IPFS-content addressed, Versioned, P2P File system," *arXiv preprint arXiv:1407.3561*, 2014, accessed on July 2022. [Online]. Available: https://arxiv.org/abs/1407.3561

[118] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," ser. Information Security and Cryptology-ICISC 2011: 14th International Conference, Seoul, Korea, November 30-December 2, 2011. Revised Selected Papers 14. Springer, 2012, pp. 406–418.

[119] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.

[120] Q. Huang and H. Li, "An Efficient Public-Key Searchable Encryption Scheme Secure against inside Keyword Guessing Attacks," *Information Science*, vol. 403, no. C, pp. 1–14, 2017.

[121] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," ser. IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10), Genoa, Italy, 2010, pp. 253–262.

[122] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[123] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," ser. Annual International Conference on the Theory and Applications of Cryptographic Techniques – Advances in Cryptology (EUROCRYPT'05). Aarhus, Denmark: Springer, 2005, pp. 457–473.

[124] Y. Lindell, *How to Simulate It – A Tutorial on the Simulation Proof Technique*. Cham: Springer International Publishing, 2017, pp. 277–346. [Online]. Available: https://doi.org/10.1007/978-3-319-57048-8_6

[125] Y. Zhang, J. Katz, and C. Papamanthou, "All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, pp. 707–720.

[126] H. S. Heaps, *Information Retrieval, Computational and Theoretical Aspects*. Academic Press, 1978.

Publications from this Thesis Work

Journal Papers

1. Abhishek Bisht, Ashok Kumar Das, Dusit Niyato, and YoungHo Park. "Efficient Personal-Health-Records Sharing in Internet of Medical Things using Searchable Symmetric Encryption, Blockchain and IPFS," in ***IEEE Open Journal of the Communications Society***, Vol. 4, pp. 2225-2244, 2023, DOI: 10.1109/OJ-COMS.2023.3316922. (2022 SCI Impact Factor: 7.9)

2. Abhishek Bisht, Ashok Kumar Das, and Debasis Giri. "Personal Health Record Storage and Sharing using Searchable Encryption and Blockchain: A Comprehensive Survey," in ***Security and Privacy (Wiley)***, 2023, DOI: 10.1002/spy2.351. (2022 SCI Impact Factor: 1.9)

Other Publications

Journal Papers

1. Sudeep Ghosh, SK Hafizul Islam, Abhishek Bisht, and Ashok Kumar Das. "Provably Secure Public Key Encryption with Keyword Search for Data Outsourcing in Cloud Environments," in ***Journal of Systems Architecture (Elsevier)***, Vol. 139, Article No. 102876, pp. 1-17, 2023, DOI: 10.1016/j.sysarc.2023.102876. (2022 SCI Impact Factor: 4.5)

2. JoonYoung Lee, MyeongHyun Kim, KiSung Park, SungKee Noh, Abhishek Bisht, Ashok Kumar Das, and Youngho Park. "Blockchain-based Data Access Control and Key Agreement System in IoT Environment," in ***Sensors***, Vol. 23, No. 11, Article ID: 5173, pp. 1-19, 2023, DOI: 10.3390/s23115173. (2022 SCI Impact Factor: 3.9)