Ancile: Privacy-preserving Framework for Access Control and Interoperability of Electronic Health Records Using Blockchain Technology

Gaby G. Dagher*
Boise State University
gabydagher@boisestate.edu

Jordan Mohler
University of Denver
jordan.mohler@du.edu

Matea Milojkovic
Winthrop University
milojkovicm2@winthrop.edu

Praneeth Babu Marella
Boise State University
praneethbabumarella@u.boisestate.edu

Abstract

Despite an increased focus on the security of electronic health records and an effort by large cities around the globe to pursue smart city infrastructure, the private information of patients is subject to data breaches on a regular basis. Previous efforts to combat this have resulted in data being mostly inaccessible to patients. Existing record management systems struggle with balancing data privacy and the need for patients and providers to regularly interact with data. Blockchain technology is an emerging technology that enables data sharing in a decentralized and transactional fashion. Blockchain technology can be leveraged in the healthcare domain to achieve the delicate balance between privacy and accessibility of electronic health records.

*Corresponding author

In this paper, we propose a blockchain-based framework for secure, interoperable, and efficient access to medical records by patients, providers, and third parties, while preserving the privacy of patients' sensitive information. Our framework, named Ancile, utilizes smart contracts in an Ethereum-based blockchain for heightened access control and obfuscation of data, and employs advanced cryptographic techniques for further security. The goals of this paper are to analyze how Ancile would interact with the different needs of patients, providers, and third parties, and to understand how the framework could address longstanding privacy and security concerns in the healthcare industry.

Keywords: blockchain, Ethereum, smart contracts, healthcare, access control, information security, smart cities

1. Introduction

In the last decade, large cities across the globe have begun to integrate technology for daily management. These smart cities utilize cutting edge developments in technology. The United States has always been a leading developer and the healthcare industry has consistently been on the forefront of utilizing those developments. In fact, the use of electronic health records (EHR) can be dated back to as early as the 1960s [1]. Unfortunately, as technology has advanced, so too have the techniques used to violate digital privacy and security. The healthcare industry, in particular, has been a major target for information theft as health records often contain private information such as the names, social security numbers, and addresses of patients. The Health Information Technology for Economic and Clinical Health (HITECH) Act, as part of the American Recovery and Reinvestment Act of 2009, sought to remedy the inadequacy of healthcare data security [1], but failed to sufficiently address the problem. In 2015, 78.8 million patients, nearly a quarter of the U.S. population, had their information stolen after a hack occurred on the insurance corporation Anthem [2]. As of June, attacks in 2017 could have affected just under 2.6 mil-

lion people, according to the U.S. Department of Health and Human Services Office of Civil Rights [3].

Theft of EHRs is rapidly becoming ubiquitous as a result of weak security systems and policy enforcement. EHRs are usually managed by individual providers, meaning all private records are stored in databases maintained by the provider responsible for creating the document. This presents several security, privacy, and control problems that emerging smart cities must resolve. First, hospitals have been known to improperly secure sensitive data. For two years, Independent Study Evaluators [4] conducted research on the vulnerabilities of hospital security. The group was able to successfully access and alter the databases of multiple healthcare facilities across the United States. Second, because providers are solely responsible for maintaining the records, data integrity can be difficult to confirm in the event that a malicious entity alters the single copy of the record. This also means that if a record is removed from a provider database, the information can be permanently lost. This flaw could have extreme consequences for the health of patients. Finally, efforts to address these problems, particularly guidelines written in the Health Insurance Portability and Accountability Act (HIPAA) of 1996, have been numerous, but ultimately unsuccessful and have limited the access patients have to their own data.

Despite flaws in its execution, HIPAA serves as an important guideline for this research. Without HIPAA compliance, it would be infeasible to implement a new EHR management system. HIPAA has four main goals: protect the right to transfer or change health insurance during a job change, reduce fraud and abuse, mandate standards for digital billing, and require the privacy and security of protected health information (PHI) [5]. To address the four different goals, HIPAA is divided into five different Titles. For the purposes of this research, only Title II is relevant, as it pertains to the security and privacy of EHRs. The sections that will be focused on are 45 CFR Part 160 and 164 Subparts A, C, and E, which consist of the Standards for Privacy of Individually Identifiable Health Information (Privacy Rule) and the Security Standards for the Protection of Electronic Protected Health Information (Security Rule) [6][7].

The intent of the Privacy Rule is to protect the privacy of patients during the disclosure of PHI. The Privacy Rule defines standards for sharing PHI with 'covered entities' and transparency with the patient on how their data is used. Creating systems that maintain compliance under the Privacy Rule requires a delicate balance between upholding privacy while allowing for the transfer and sharing of health information as it pertains to giving high-quality care to patients and society as a whole [6]. Similarly, the Security Rule defines specific measures that must be taken to uphold the Privacy Rule and protect electronic health information. According to the U.S. Department of Health and Human Services [7], the Security Rule "operationalizes the protections contained in the Privacy Rule by addressing technical and non-technical safeguards".

In context of existing healthcare data systems, improvements must be made to reach even bare minimum compliance with the Privacy Rule and Security Rule. Thus, our research seeks to accomplish four goals while achieving HIPAA compliance: give ownership and final control of EHRs to the patient, securely control who can access documents and track how records are used, allow for secure transfer of records, and minimize ability for unauthorized actors to derive PHI.

The HIPAA Privacy and Security Rules establish significant requirements for the transfer of EHRs within the United States. Similarly, standards like those proposed by Health Level Seven International (HL7) [8] dictate international standards for the transfer of EHRs. HL7 far expands beyond the scope of the HIPAA Security and Privacy Rules by creating standards applicable to all stakeholders involved in the health industry, including those not related to human treatment like veterinary drug companies. As a result, HL7 provides a robust analysis of the correct way to handle EHRs. HL7 outlines 5 goals for EHR management systems: interoperability, security, quality and reliability, efficiency, and understandability [8]. Ancile was designed primarily to meet the requirements of HIPAA; however, both Ancile and HL7 strongly prioritize the need for increased interoperability. As a result, Ancile advances the primary goal of HL7 and can be adapted in the future to cover the additional standards

of HL7 should it be used internationally.

To accomplish the four goals outlined, we have designed a framework using blockchain technology. A blockchain is an append-only data structure that functions as a distributed ledger. This is accomplished by replicating all the data on the blockchain across all nodes in the system. As a result of this redundancy, a blockchain is easily verifiable and has no single point of failure [9]. Blockchains are created by having nodes in the system 'mine' blocks, or create additions to the structure using a hash of transactions that people have recorded on the blockchain. This structure makes blockchains immutable unless participating nodes with 51% of computation power on the blockchain choose to rewrite the chain. Both mining and retaining so many copies of the same data does have its costs in computational power and storage, but they are necessary for a blockchain to be a completely decentralized, immutable system [9].

The most famous example of blockchain technology is the Bitcoin [10] blockchain. Since its proof of concept in 2009, Bitcoin [10] has grown into a blockchain for cryptocurrency with which millions across the globe participate. Bitcoin was unrivaled in popularity, but since its creation a number of other blockchains have also garnered substantial public attention. Our framework is based on the Ethereum [9] blockchain. Similar to Bitcoin, Ethereum [9] can also be used for cryptocurrency; however, unlike Bitcoin, Ethereum includes additional functionality like the ability to use 'smart contracts'.

Smart contracts are functions that can be written to the blockchain and then executed by all nodes on the block. Ethereum [9] is able to manage these functions by requiring people to pay 'gas' in the form of Ether, Ethereum's currency, if they want to run a contract. Because of the gas cost, people cannot infinitely run programs on nodes in the system. Additionally, Ethereum also allows users to create private or permissioned blockchains that can be managed and controlled by a smaller set of users. Some argue that private and permissioned blockchains are counter-intuitive to the goal of decentralization; however, they present advantages like increased privacy control and the ability to modify the gas cost requirement [11].

The use of a permissioned blockchain for EHRs can also be found in [12]. Our framework uses other similar concepts to those in [12] and [13], specifically for storage and access control. Like Ancile, these applications can be integrated with a providers existing system. Furthermore, the applications also establish access controls that allow patients greater control over their medical data, especially in the process of transferring medical documents from one provider to another. In regard to storage, both [12] and [13] refrain from storing entire records on the blockchain. [12] stores hashed pointers to medical records and permissions, while [13] stores indices to records on the blockchain. By implementing this type of storage concept, the scalability of a system is expanded.

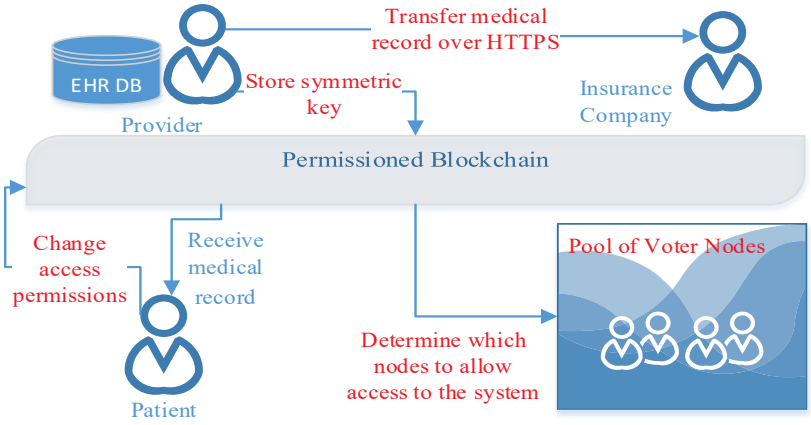


Figure 1: This image demonstrates an overview of how different parties may interact with each other using Ancile. Including three different parties, the patient, provider, and insurance company, this diagram briefly relates each party to each other and the blockchain. The red text indicates the unique differences between Ancile and previously proposed blockchain health data management systems and blue text represents already proposed or existing processes in EHR management.

1.1. Contributions

Figure 1 outlines Ancile's ability for multiple parties to securely interact with the blockchain and its information. As a result of Ancile's prioritization

of secure interaction, the framework we propose includes several contributions designed to increase privacy and interoperability. First, unlike previously proposed blockchain EHR systems, the Ancile blockchain stores hashes of the data references while sending the actual query link information in a private transaction over HTTPS. JP Morgan's Quorum [14] also uses private transactions for privacy but lacks techniques like proxy re-encryption, which our framework employs to streamline the secure transfer of EHRs. Additionally, the use of proxy re-encryption allows us to store keys and small encrypted records directly on the blockchain, easing the transfer of records like prescriptions to pharmacies or other third parties. This also ensures that users have no need to locally store keys, enabling patients to remove access permissions if desired.

Second, our design focuses on the ownership rights of the patient. As a result, our design commits to the idea that the data is owned by the patient and not a currency to be exchanged. Thus Ancile does not include any form of mining incentive beyond simple use of the system. We assume that providers and governments already have an incentive to secure the medical information of patients. Furthermore, we utilize smart contract functionality for access control to account for the varying roles of patients, providers, and third parties on the blockchain. This allows for a stratification of roles that can suit the different needs of users. For example, in the event that a patient is a minor, we maintain the ownership of the data by the patient while allowing for controls to be put in for parents or guardians.

Finally, we base our permissioned blockchain structure on a consensus algorithm rather than proof of work. This allows us to have increased validation when adding nodes to the network or removing harmful users.

1.2. Technical Difficulties

Blockchain technology offers many advantages for an EHR management system, but there are also inherent limitations. The primary limitation of blockchain technology is that a collusion of 51% of mining nodes on the system could result in the rewriting of the chain structure [9]. Thus, to achieve the

advantages of a decentralized system, participants must have some trust that at least 50% of mining nodes would not want to violate the immutability of the blockchain. Secondarily, using a permissioned blockchain mitigates the ability of external actors to gain access to PHI, but cannot mask the record of transactions. This allows nodes to conduct unfavorable network analysis. By analyzing transactions on the blockchain, an adversary may be able to determine the frequency with which a specific node visits a physician or the providers or third parties with which a specific node associates. Finally, because blockchains are distributed systems, there is a high storage cost for their operation [9]. As a result, large data cannot effectively be stored on the blockchain. Thus, while blockchains can be used for access control and data integrity, the data itself must be stored elsewhere and could be vulnerable to attack entirely separate from the blockchain.

2. Related Works

In regards to implementing the blockchain in the healthcare industry, a few research articles in particular detail potential systems available for healthcare providers. The major objective of these articles is to introduce blockchain systems that would establish access controls for medical data and allow patients greater oversight of their personal medical information. In Alexander Samarin's article [15] medical records are privately stored on the cloud and solely accessible by the patient. While this clearly transfers full ownership of medical records to patients, it neglects to account for the need to share medical information with multiple entities. Moreover, it cannot address the situation in which a doctor must keep a patient's medical records undisclosed even from the patient, such as psychotherapy notes.

Our framework utilizes storage techniques similar to the concepts proposed by Ariel Ekblaw et al. [12], Laure Linn and Martha Koo [13], Drew Ivan [16], and Brodersen et al. [17] which continue to store patient data in a provider's existing database while integrating the blockchain as an access control layer. Although

medical records continue to be stored in providers' databases, [12][13][16][17] expand patient control over personal medical records by requiring that patients approve the transfer of their records to other providers or third parties.

To increase the scalability of their schemes, [12][13][17][18][19] do not store entire patient records on the blockchain. For instance, in [13] an index containing hashed pointers to a patient's records as well as encrypted data is stored on the blockchain. Similarly, the blockchains in [12][17] store only hashed pointers to records and permissions. [18][19] only include Uniform Resource Locator (URL) references to medical records and the data necessary to run smart contracts on the blockchain. Ancile likewise continues to store most patient data in providers' existing databases, while storing references to the data and access permissions on the blockchain. The ability for patients to have greater control over their personal health records without requiring them to store and manage the actual records enhances the scalability of these schemes.

Medical record privacy, specifically access control, is a principal concern in the health care industry. [12] suggests implementing a permissioned blockchain structure and encrypting data off the blockchain. [18] implements a permissioned blockchain in which only providers, health plans, and governmental parties participate. Our framework also utilizes a permissioned blockchain structure, but allows patient interaction with the blockchain while giving clear levels of authority to each node. This is consistent with our goal to broaden patients' control over their medical records while maintaining access controls.

To preserve privacy, healthcare blockchains must defend EHRs from threats attacking various portions of the system. For example, man-in-the-middle attacks can compromise a system's access controls. [13][15][17][18][16][19] propose varying techniques to establish and secure access controls. [13] proposes biometric identity systems for authentication of parties. [19] recommends encrypting pubic data with a network-wide symmetric key and private data with secret keys. Furthermore, [15] addresses the issue by employing a 'deposit box' when a record is transferred to another party. Using this approach, once a patient authorizes the transfer of a personal medical record, the record is copied into

the deposit box for a specified time period and the recipient is given access to the record for that time period. To further authenticate an individual's identity on the blockchain, [17] suggests trusted authorities such as banks and employers provide verification of an individual's identity in the system. Likewise, Ancile applies a consensus verification process before registering nodes.

Data forensics is an additional threat to medical data. Unfortunately, it is an issue inherent in blockchains. To increase the difficulty of forensics, [18] proposes intermingling fictional records with valid records on the blockchain to ensure data privacy; however, this assumes a reasonable method for authorized parties to sift through records. [16] proposes three different options for the transfer of medical records to mitigate the threat of man-in-the-middle attacks: providers' databases can be directly connected to the blockchain, providers' existing systems may send records to the blockchain, or providers may send the records to patients who then manually upload the records to the blockchain. [16] notes the third option assumes patients would not withhold any medical information from the blockchain. Alternatively, our design handles access controls by encrypting data on the blockchain and implementing various smart contracts.

Another way to improve access control is the use of private transactions. [14][20][21] are three such systems that utilize these types of transactions. In the Quorum system [14], public transactions are validated by every node on the blockchain while private transactions are stored off-chain and only validated by nodes party to the transaction. Due to this, the public state record is identical for all nodes but the private state record varies among nodes. IBM's Hyperledger [20], as well as [14], encrypts private data in transactions to ensure only authorized entities are able to view confidential information. Similar to [14][20], nodes in Hearn's Corda [21] system only reach consensus on public states and private states to which they are party. [21] also applies cryptographic hashes to data in transactions for data integrity. Similarly, our framework encrypts and hashes data stored on and off the blockchain.

In advocating for the importance of transaction privacy on the blockchain, [17] states transactions must not be traceable back to specific patients. To do so, [17]

suggests implementing tokenization: a process through which only a reference to sensitive information is public, allowing the raw, confidential information to be kept private. Furthermore, [17] specifies the need for the secure storage of medical records off the blockchain. Because the blockchain is often used simply to store references to records that are stored in an external database, the database itself must be secure. By encrypting data in the database, [17] and our design safeguard data both on the blockchain and in the database.

In order to add new blocks to the private or permissioned blockchain, [14] implements an algorithm named QuorumChain in which a specific number of nodes are given the ability to vote for which blocks to add to the blockchain. The QuorumChain algorithm is contained in a smart contract, simplifying the voting process. Our design utilizes this concept in the mining process and other blockchain maintenance.

3. Preliminaries

3.1. Blockchain Mining

In order to reach consensus on the order of transactions in a trustless system, blockchains implement two concepts: proofs and miners. In a permissioned blockchain, specific nodes in the network are designated 'miners' [11]. The role of miners is to validate transactions in a block and append the block to the blockchain. In order to do so, first miners choose transactions from a pool, locate the hash of the previous block, and choose a random number called a nonce [9]. The transactions and previous block hash are then hashed together. If the hash does not solve a computational problem, the nonce is incremented and the contents are rehashed until the problem is solved. Miners are then rewarded for finding the next block. Rewards are typically monetary, as is the case with Bitcoin, but may vary depending on the purpose of the blockchain.

Once a new block is found, it is added to the blockchain, usually in the form of a tree. The Bitcoin blockchain specifically uses a Merkle Tree [10]. A Merkle Tree is constructed as a binary tree, the base of the tree containing the data

while each parent node is the hash of its two children [22]. Because Merkle Trees utilize hashes, if any part of the data is changed, all the nodes on the network would notice the alteration.

Different blockchains utilize varying types of proofs such as Proof-of-Work and Proof-of-Stake to determine which miner's block will be appended next [11]. Likewise, the QuorumChain Consensus algorithm [23] can be used to determine the next block. The QuorumChain Consensus algorithm [23] relies on a majority-based voting method to achieve consensus. The algorithm is executed through a smart contract which tracks votes and eligible voter nodes. Once a possible new block is found, voting nodes may then vote on the next block. In order to be appended to the blockchain, the block must receive the majority of the votes and the number of votes received must be above the threshold amount [23]. This is to prevent adding multiple blocks with the same transactions. To decrease the likelihood of multiple miner nodes creating identical blocks at the same time, the QuorumChain Consensus algorithm implements timeout sessions, a pseudo-random amount of time that must elapse before a miner node can create a block.

3.2. Eth_Calls and Internal Transactions

To reduce the number of transactions stored on the blockchain, the Ethereum Blockchain uses '*eth_calls*' and 'internal transactions.' Similar to transactions, *eth_calls* may interact with smart contracts [1]. However, unlike transactions, *eth_calls* are only executed by the local node and are therefore not stored on the blockchain. They may be seen as simulations of transactions that are capable of sending return values to the node. Like transactions, *eth_calls* must be sent by an external actor [24]. Alternatively, internal transactions occur when smart contracts interact with each other after being initiated by a transaction. Similar to *eth_calls*, these types of transactions are not stored on the blockchain. *Eth_calls* and internal transactions greatly increase the scalability of blockchain-

[1] JSON RPC API: https://github.com/ethereum/wiki/wiki/JSON-RPC

based systems by reducing the amount of information stored on the blockchain.

3.3. Proxy Re-Encryption

Proxy re-encryption solves the issue of transferring encrypted records between nodes without sharing symmetric keys by using a proxy. The proxy is responsible for reconstructing an encrypted message in such a way that another user could use their private key to decrypt the document, even if it was not originally encrypted with their associated public key [25]. This system allows secure sharing between parties without fully decrypting the document during transfer process, as shown in Figure 2.

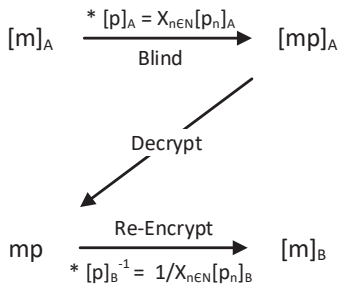


Figure 2: The process for blinded re-encryption based on the protocol defined in [26].

In this paper, we utilize a distributed proxy re-encryption scheme with blinding, where multi-parties (proxies) partake in the re-encryption process [26]. To do this, a message is encrypted with a master public key, and the associated private key is then distributed in pieces to the proxies. In doing so, the proxies can re-encrypt the message while unable decrypt the full message. To further prevent proxy nodes from accessing the message, a blinding re-encryption scheme, like [26], will use homomorphic multiplication to create an encrypted blind value from random numbers chosen by each proxy. The message is then homomorphically multiplied by the blind value, thus creating a plaintext message that is obscured unless the blind value can be determined [26]. Thus, distributing

the private key and blinding the message, the message can only be decrypted by the intended receiver or after every proxy agrees to collude.

3.4. Types of Nodes

As a blockchain continues to grow, the scalability of the system can be compromised, because only users with large storage spaces and computational power will be able to partake in the blockchain as miners or full nodes. To circumvent this issue, the blockchain supports three different types of nodes: full nodes, light nodes, and archive nodes [27]. Full nodes process every transaction and store every block in the blockchain. On the other hand, light nodes only store block headers, which includes the hash of the previous block, the hash of the Merkle Root, and the nonce. By storing the block header, the light node can verify certain transaction have not been altered, without committing large portions of memory to the blockchain. Light nodes also have the ability to access specific data they desire.

Similar to full nodes, archive nodes store every transaction and block on the blockchain [28]. Additionally, archive nodes store transaction receipts and the entire state trie [29]. Using this information, archive nodes are able to help the network retrieve needed data [28]. The versatility of these three different types of nodes increases the scalability of the Ethereum Blockchain such that large corporations and individual users are able to interact with the blockchain for their respective purposes and with their available resources.

4. Proposed Framework: Ancile

4.1. Overview

Our proposed framework, Ancile, uses six unique types of smart contracts for operation: Consensus, Classification, Service History, Ownership, Permissions, and Re-encryption. By using six separate contracts, we enable patients to benefit from increased utility while minimizing the need to interact with every contract. This improves the efficiency of the patient experience and reduces

privacy threats. To create a high level of separation, we use the contracts to generate other contracts. In doing so, the patient can be the only node expressly given the location of their information.

Using smart contracts, Ancile maintains cryptographic hashes of stored records and query links, confirming the integrity of EHR Databases. Patients can also view and control who has permissions for their private information by using a smart contract to manage access control. Moreover, patients may give transfer permissions to other nodes. This is possible through the use of identity-checking, to confirm who may access records, and proxy re-encryption, to avoid having to re-encrypt the record for each transfer. Furthermore, by sending the query links for the records securely off of the blockchain, Ancile ensures that the three tools required to access an EHR, the encrypted record, the query link, and the symmetric key, are in different locations. In the following sections we outline specifically how Ancile would be maintained, the different software modules needed for client use, the specific role of each smart contract, and the architecture of the framework.

4.2. Software Components

Ancile consists of three main software components: *Database Manager*, *Cipher Manager* and *Ethereum-Go Client*.

A. Database Manager. Ancile incorporates EHR Databases by generating query links to records stored in the existing system. The Database Manager is used to navigate existing EHR Databases and for generating the link that maps to a record. Moreover, the Database Manager will also create hashes of both the record and the query link to place on the blockchain. Using existing databases risks actors altering or removing data directly from the database, subverting the blockchain; however, the hashes can be used to confirm data integrity. If data is not returned, the hashes can be used to confirm what specific data has been lost, creating an indisputable record should additional action need to be taken.

B. Cipher Manager. The Cipher Manager is responsible for all cryptography in Ancile. Decryption of all files, three different forms of encryption, and

the storage of encrypted data is completed using the Cipher Manager. The first form of encryption is symmetric key. Ancile uses symmetric key encryption on larger files because it is efficient and eliminates the need to re-encrypt the files later. The second form of encryption the Cipher Manager uses is public key. Public key encryption is used to protect information during distribution and to clearly indicate who may access PHI. The third form of encryption managed by the Cipher Manager is proxy re-encryption. When bestowing record access to a third party, proxy nodes may use their Cipher Managers to re-encrypt copies of the symmetric keys stored on the blockchain. This vastly expedites the transfer process as compared to current methods. Finally, the Cipher Manager is responsible for decrypting all encrypted information retrieved from Ancile.

C. Ethereum-Go Client. The Ethereum-Go Client [27], sometimes called Geth, is the main Ethereum CLI client written in the Go programming language. Geth is an access point to Ethereum networks, including the public, test, and private Ethereum networks. Ancile is designed to function on a permissioned Ethereum blockchain; thus, the Geth client would be used by permitted nodes to access the private blockchain. Having the Geth client would signify to Ancile that a particular system is a node. Additionally, Geth [27] may be accessed using JSON RPC endpoints on the Internet. These nodes may be full or light nodes, allowing for versatile roles of patients, providers, and third parties. Users may access their node's information with Geth on the client side using a 'wallet' [30]. The functionality of wallets may vary depending on the type of node. By using wallets, users may access their node's information over HTTPS [30]. As a result, the Geth client allows Ancile to have a user-friendly interface that can be accessed on the web and adapted as needed.

4.3. Smart Contracts

Ancile consists of six smart contracts: *Consensus Contract*, *Classification Contract*, *Service History Contract*, *Ownership Contract*, *Permissions Contract* and *Re-encryption Contract*.

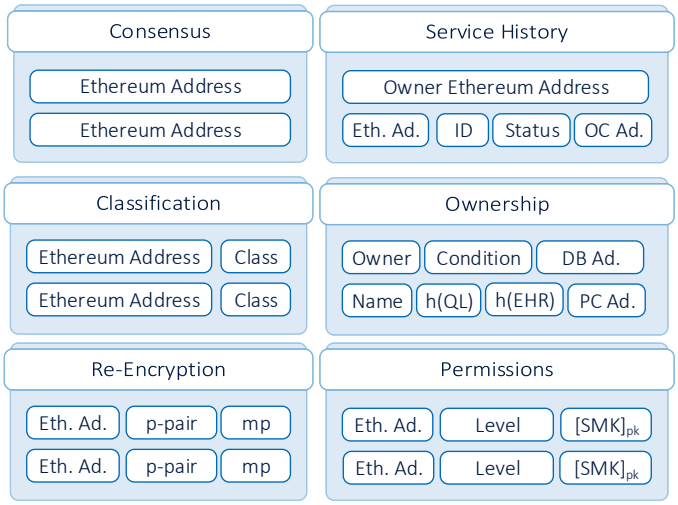


Figure 3: This image displays the memory fields that each smart contract would be responsible for tracking. QL represents a query link, while SMK and PK represent symmetric and public keys respectively. An expression preceded by an 'h' and enclosed in parentheses represents hashed information, and brackets represent encryption using the key indicated by the following subscript.

4.3.1. Consensus Contract

The Consensus Contract (COC) is a global contract responsible for maintaining blockchain mining, user registration, and certain overwrite procedures. As seen in figure 3, the COC stores the Ethereum addresses of nodes with voting permission. For mining, the COC operates using the QuorumChain [23] Consensus algorithm.

For user registration, the COC is used to validate nodes that request higher levels of classification when added to the system. The use of consensus ensures existing nodes are constantly confirming that new nodes do not pose a threat to the system. It should be noted that during the beginning stages of Ancile, the COC would be empty. Thus, a temporary administrator node would need to add initial nodes, such as long-confirmed providers and third parties. Once there are enough full nodes to allow the removal of the temporary administrator, the consensus process would be implemented.

Finally, the COC may also be used for overwriting nodes that cease to exist or that have been deemed harmful to the system. An example of this is when an insurance company goes out of business, the permissions of the associated node need to be revoked. To do this, a request may be submitted by a voting node, and the rest of the nodes must reach a majority to remove the node from the system. This would then entail overwriting their classification as terminated, removing them from the COC, and deleting their information from various PCs and OCs.

4.3.2. Classification Contract

The Classification Contract (CLC) is responsible for classifying the different levels of nodes in the system as patients, providers, or third parties. There is only one CLC for the entire blockchain to utilize. As seen in figure 3, the CLC maintains two data fields, Ethereum addresses of all nodes and their associated classifications. With this information, the CLC can confirm what nodes are already registered in the system to prevent double registration. Moreover, because the COC is used during the node registration process to determine node classification, the CLC can be used to verify the identity of a node without repeating the consensus process. Thus, the use of the CLC reduces the complexity of access control in later contracts by giving a single point of reference that is confirmed by the consensus process.

4.3.3. Service History Contract

The Service History Contract (SHC) maintains the relationship histories of nodes. A new SHC is created for every node during the registration process. As seen in figure 3, the SHC stores the Ethereum address of the patient, the Ethereum addresses of all related nodes, their associated IDs, a relationship status, and an address to the applicable Ownership Contracts (OC). The status field may indicate an active or inactive relationship. When nodes are interacting with Ancile, their SHC provides them with a comprehensive list of their healthcare relationships, previous and current. Additionally, the use of IDs in the SHC

allows providers to identify patients, or vice versa, using existing IDs. The SHC is also responsible for querying the patient for permission confirmation. When a provider wants to establish a relationship with a patient, the SHC of the patient will request permission before updating. This is significant because the permissions function ensures the patient is always aware of their relationships.

4.3.4. Ownership Contract

The OC is responsible for tracking the records that providers store for patients. An OC is created when a new relationship is formed between two nodes. As seen in figure 3, the OC contains many data fields with varying purposes. The OC can be identified by the Owner field, which signifies which patient owns the records listed. Condition and Date fields may follow the Owner field. They indicate if there are special conditions for the owner. For example, a parent or guardian may be the owner of a minor's data until they come of age. The Date field would then indicate to the OC that ownership should be transferred at that time. The Condition field may also indicate that there are shared owners of the information, which may be the case in a provider-provider relationship. Additionally, the OC contains the information needed for the patient node to find the EHR Database of a provider. The OC then lists each patient record with a filename, a hash of the file's query link, a hash of the record itself, and an address to a Permissions Contract (PC).

The two hash fields are critical to establish data integrity. Storing records in provider EHR databases allows us to minimize the storage requirements of the blockchain and also makes use of the existing system; however, as a result, the provider can make alterations to a record without indicating the changes on the blockchain. Thus, the hashes can be used to confirm that no changes have been made off of the blockchain, as the hash is unique to the original record. Moreover, Ancile stores a hash of the query link rather than the query link itself to ensure the query link, key, and record are always in different locations. Instead query links are sent over HTTPS and the hash in the OC can be used to confirm that the link has not been altered during the transfer process.

4.3.5. Permissions Contract

The Permissions Contract (PC) is specific to every record and is built by the OC when a new record is added to the system. As seen in figure 3, it is designed to store the Ethereum addresses of all nodes that may interact with a record, a level of access, and a symmetric key encrypted with the public key of each node. The patient, the provider of origin, and a Re-encryption Contract (RC) will be automatically written to the PC with Owner, Read, and Blind level access respectively. In the case of special exceptions, such as psychotherapy notes, the provider may indicate in the transaction that the record should be kept private from the patient. The provider would then be written with Owner level access and the patient is given Blind access. The different levels of access are as follows:

- Read- A node has a symmetric key generated for them when the record is first added or through proxy re-encryption.
- Transfer- A node may add other nodes with Read access. When a node is given this level, special conditions may also be added. For example, a provider node may be given permission to give Read access only to other provider nodes.
- Owner- A node has full control of the PC. They can add other nodes of any level of access, remove nodes from the PC, and alter the levels of access for any existing nodes.
- Blind- A node is only given the address of the PC. This level would typically be used for patients who should be able to see who can view their records, but may not be allowed to view the records themselves. It is also used to designate that the RC may retrieve the symmetric key encrypted for the proxy nodes.

4.3.6. Re-encryption Contract

The Re-encryption Contract (RC) is used for proxy re-encryption. In Ancile, proxy re-encryption on the blockchain requires that a set group of proxy nodes be given a master public key with a shared private key. A RC should be created

every time a new set of proxy nodes is established. Using a high number of proxy nodes per set enables re-encryption schemes to be effective while ensuring that the likelihood of proxy collusion is low. In Ancile, each proxy will be responsible for choosing a blind value, p, encrypting it, and decrypting blinded message portions on their own systems. When the proxies need to combine their values, they will each send their contribution to the RC.

As seen in figure 3, the RC stores the proxy node's address, the encrypted pairs of p values, and the plaintext blinded message. This is a result of the RC employing homomorphic encryption to generate those values. At this time, smart contracts support is limited to 256 bit unsigned integers [31]. Symmetric keys, which will be the messages to re-encrypt, are often 128 bits or greater in length. Once encrypted, the symmetric key would be of an even greater size. Thus, in order to use homomorphic multiplication in the RC, the symmetric keys must be kept at a smaller size until developments in Ethereum allow smart contracts to support greater values.

4.4. Framework Architecture

The following diagrams demonstrate the architecture of Ancile by assessing how the framework would be used in various situations. The framework uses four distinct forms of actions, as seen in figure 4. The first action, represented in solid blue, is a standard blockchain transaction. These are written to the blockchain and mined using the QuorumChain consensus algorithm. The second action, represented in dashed blue lines, represents internal transactions.

The third type of action, represented in orange, is an *eth_call*, which is used when data needs to be sent to a smart contract, but does not need to be written to the blockchain. This allows for a more efficient and private system while employing the functions needed to operate Ancile. The final type of action, represented in gray, is a non-blockchain action. This could represent data being transmitted over HTTPs or something happening internally to a node. The use of non-blockchain actions may also represent the use of private transactions. A private transaction uses an external method to transmit sensitive data, while

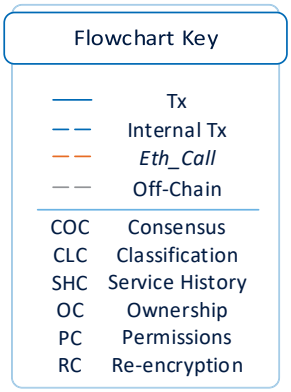


Figure 4: This image displays a key for interpreting the different forms of action and abbreviations used in the framework architecture.

placing a hash of that data on the blockchain. As a result, only those who can recreate the hash can validate the legitimacy of the transaction. Using private transactions preserves the data integrity offered by the blockchain, while increasing privacy.

4.4.1. Adding a Node

The process for adding a new node can be seen in figure 5. The image depicts a patient being added to the blockchain, but it should be noted that a similar process would occur for any classification of node. It should be assumed that new users have created wallets and received an Ethereum address prior to this process. It should also be assumed that providers and third parties possess a public identifier that is unique to their organization. An example of this identifier may be the provider numbers assigned by the federal government for Medicare purposes [2]. Additionally, because patients are often already represented with numerical IDs in existing provider systems, public IDs for the patient should be understood as that existing value, which should be kept off

[2]Hospital general information: https://data.medicare.gov/Hospital-Compare/Hospital-General-Information/xubh-q36u/data

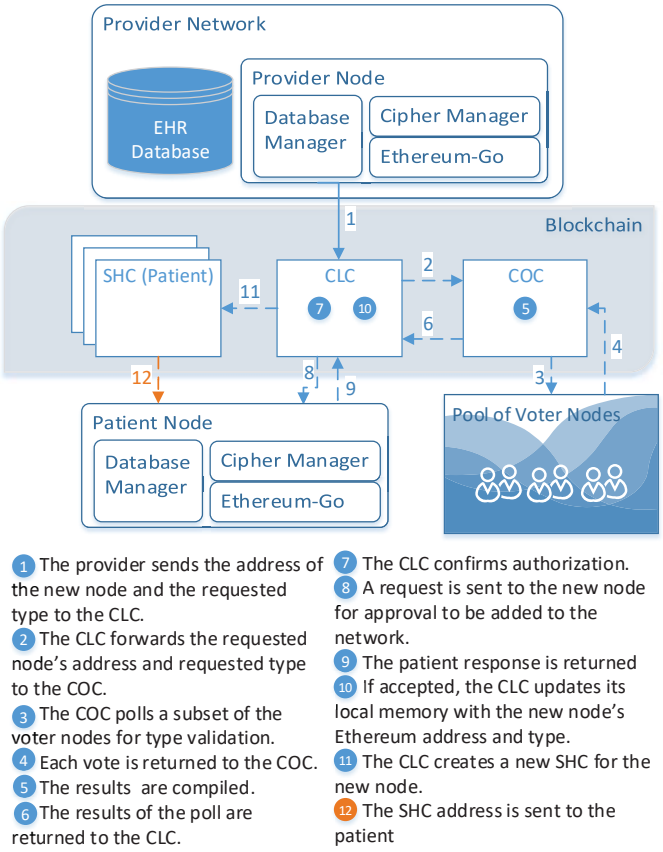


Figure 5: The process for adding a patient node to the permissioned blockchain. Assume that new nodes have created a wallet and established an Ethereum address prior to this process.

the blockchain for privacy reasons.

The process of adding a node begins by having voter nodes validate that the public ID suits the requested classification. Because adding a patient has the lowest level of permissions on the blockchain and because only a numerical ID for patients is sent to voters, patients will be added to the system with little validation. Alternatively, the validation process should be more extensive for providers and third parties. Voter nodes would be responsible for authenticating the classification request is reasonable by ensuring the node is legitimate provider or third party. This process could entail confirming the existence of a

non-registered provider matching the given ID.

This validation process, in addition to the fact that requests to add other nodes must come from already registered nodes, results in a low probability that illegitimate actors could overtake the system. After validation, adding a node to the system concludes by the CLC generating a new account and associated SHC. Patients or other users would then receive their account information from the node who put in the request, similar to how patients may make online accounts or fill out forms during the first visit to a new provider.

4.4.2. Registering a Patient

Registering a patient is one example of establishing a relationship between two different nodes. This process would be completed every time a new patient visits a provider. Using the SHC, Ancile documents each time a new relationship is formed. As seen in figure 6, registration begins by confirming that the patient is a registered node in the system. If the patient is not a node in the system, the 'Adding a Node' process would need to be completed first. After confirming node status, the provider sends the pertinent information in a transaction to their SHC.

The registration process continues by requesting the patient node validate the relationship. This gives the patient the ultimate say on the providers with which they associate. If the patient refuses the relationship, the process is terminated and the provider is notified. However, if the patient has agreed to the relationship, a new OC is created to represent the union. The OC would then automatically fill the Owner field with the Ethereum address of the patient and the database information of the provider network. Any special conditions to ownership may also be added at this step. Finally, the SHCs of both the patient and provider are updated with the OC address for future reference.

The registration process would be similar for any relationship. The use of the SHC and this registration process allows users to form relationships with a myriad of other nodes. For example, it would be employed to register a relationship between an insurance company and healthcare provider. Establishing

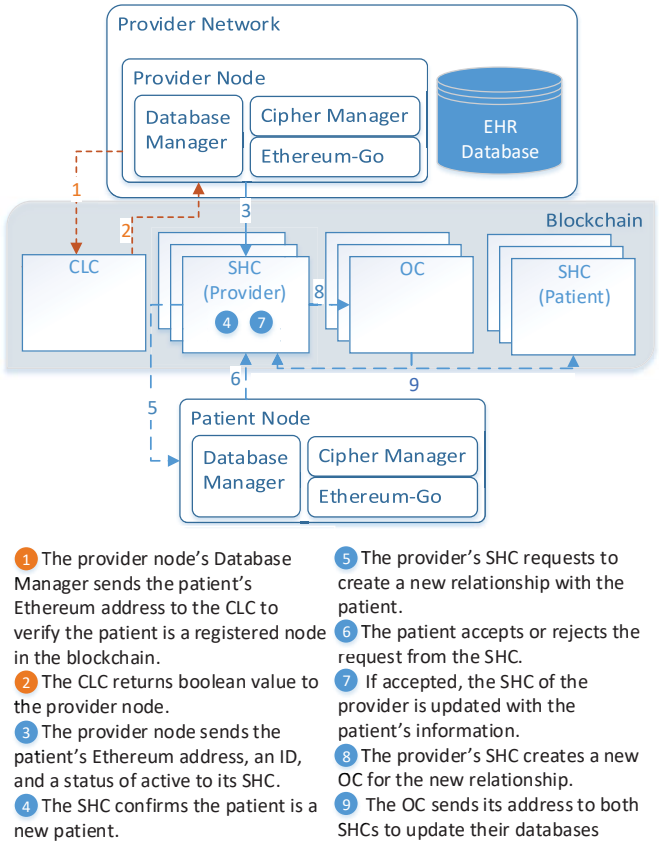


Figure 6: The process for registering a new provider-patient relationship. The provider node must know the Ethereum address and ID for the patient prior to this process; thus, it is likely this would be completed at a provider's office by the patient interacting with the provider network.

a relationship in Ancile would be preferable for any two entities that need to share protected data.

4.4.3. Changing Access Permissions

There are several situations where a patient may want to give increased control over their records to a provider or other party. Figure 7 depicts the process for giving a provider Transfer or Owner access to a record. It should be noted that during PC creation it may be preferable to give the original provider

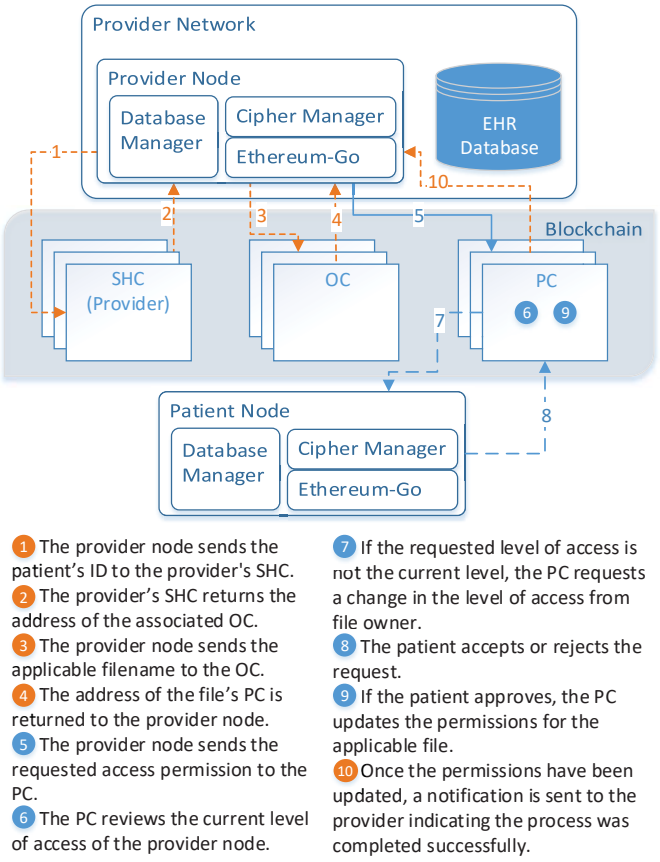


Figure 7: The process for requesting additional permissions. It should be assumed that the patient has Owner access for the requested record.

irrevocable Read access, as the record is stored on their network, but this is not required. For figure 7, it should be assumed that the patient has ownership of the record.

The process for changing access permission begins by locating the PC for the record and sending a request for permission changes. The PC will then confirm that a change in permission is possible before issuing a request to the patient. If the patient did not have ownership of the record, those with ownership would be sent the request. The PC will then update its local database and return a

positive or negative notification to the provider. The provider node would be able to scan the PC for changes, but the use of a notification is to quicken the process in the case of a time-sensitive event. This process would be repeated for any entity that may request a change in permissions.

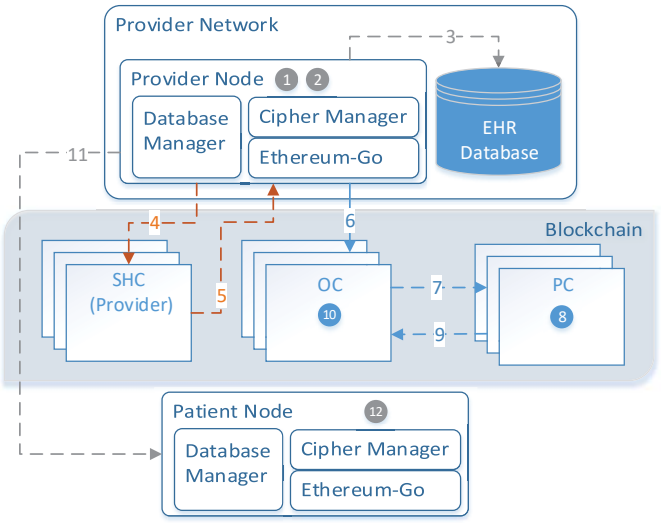
Alternatively, the situation may arise where a patient wishes to give increased permissions without a formal request. The patient would first need to know the Ethereum addresses of nodes they wish to give access. Then a patient would simply send a transaction to the PC indicating these changes. Nodes with Owner level access, may give additional permissions to any node on the blockchain. This gives flexibility for patients to indicate a spouse or other legal representative as having a right to the information.

4.4.4. Adding a Record

The process for adding a record begins with internal encryption in a provider node. It should be assumed that the provider and patient have already established a relationship and have a shared OC. Once a provider node creates a new record, it will be moved to the Database Manager and a query link to the EHR Database will be created. The Database Manager should then automatically begin generating hashes of the record and query link and communicating with the Cipher Manager to encrypt them before storing in the EHR Database. Following the encryption, the provider node will locate the OC address for the patient and upload the record hash, the query link hash, and the encrypted symmetric key to the blockchain. After uploading the pertinent information, the OC will create a new PC for the record, automatically adding permission fields for the patient, provider of origin, and RC. The query link for the record is then sent to the patient, who may access the record when desired. Figure 8 depicts the process of adding a record to an EHR Database and then using Ancile for data integrity and access control.

Additionally, Ancile may also be used to store small records. This functionality may be useful for the quick transfer of records like prescriptions. However, the records would need to be small because of the high cost for mining and

storing data. To store a small record, the same encryption process would be completed but the record would be uploaded to the blockchain rather than a hash of the record. As a result, no query link would be needed to access the record, simplifying the retrieval process. Unfortunately, all the encrypted tools needed to access the record are available to the entire system when a small record is placed on the blockchain. As a precaution, records uploaded to the blockchain should not contain information like social security numbers or home addresses.



1 The provider's Database Manager generates a query link to a free location in memory, hashes the link and the record, then sends the link and record to the Cipher Manager.
2 The Cipher Manager generates a symmetric key and encrypts the new record and link, then encrypts the symmetric key with the public keys of the provider, patient, and proxy set.
3 The Database Manager stores the record in the EHR Database.
4 The provider node sends the patient's ID to their SHC.
5 The address of the associated OC is returned.
6 The provider node sends the record name, query link hash, record hash, and encrypted symmetric keys.
7 The OC creates a new PC for the record and sends the encrypted symmetric keys to the PC.
8 The new PC auto-creates the provider, patient, and RC permissions.
9 The PC sends its address to the OC.
10 The record information is added to the OC's local memory.
11 The encrypted query link is sent to the patient over HTTPS
12 The patient node stores the query link in its Cipher Manager.

Figure 8: The process for adding a new record. The process for registering a relationship must be completed prior to the start of this process.

4.4.5. Retrieving a Record

Retrieving a record is a non-taxing process because no transactions are required. As seen in figure 9, the process begins by the patient locating the OC for the provider who stores the record. The patient then issues a request for the record. If the patient has permission to access the record, the encrypted symmetric key is returned. Once the patient decrypts the key, they may decrypt the query link they should have stored in their Cipher Manager, access the record in the provider's EHR database, and decrypt the record.

This process would require very little client effort, but may take time to retrieve and decrypt all three tools needed. However, because patients access their nodes through online wallets, they would be able to access their records on any machine with Internet connectivity. The ability for this retrieval to occur on essentially any computer, and even mobile device, vastly improves the interoperability of EHRs. The process depicted in figure 9 does not represent the process for retrieving small records that are located on the blockchain, but that process should be even simpler. The only notable difference is that the record itself would be returned in addition to the symmetric key and the patient would not need to access the provider's EHR database.

4.4.6. Transfer a Record

Smooth transferring of records is necessary for any EHR management system. Ancile uses proxy re-encryption to balance the need for accessibility while maintaining security. Figure 10 depicts the process of one provider sending a record to another. It should be noted that the process for transferring a record could technically occur by retrieving a record, decrypting, and sending to another party. Thus, Ancile cannot ensure all data movement is tracked, but can verify who is permitted to share records and with whom records can be shared. In this way, Ancile may be used as an indisputable ledger should external actions need to be taken.

The process begins by Provider A locating the necessary PC. It should be assumed that Provider A has Transfer or Owner level access, otherwise the PC

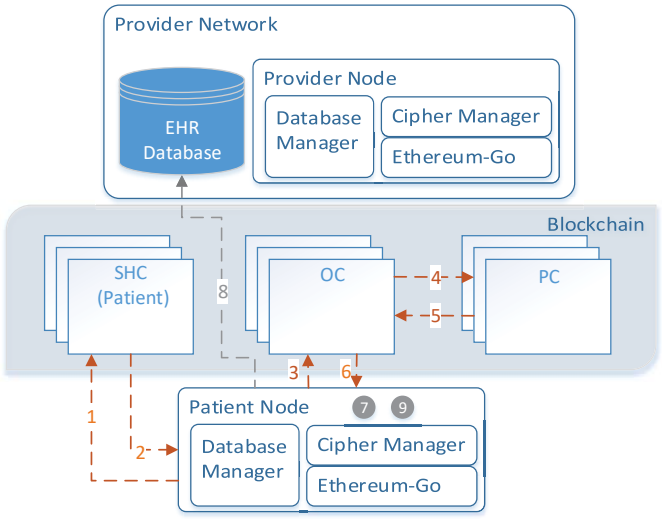


Figure 9: The process for a patient node retrieving their record. Assume the patient has at least Read access for this record and therefore has a symmetric key encrypted for their use.

would end the process. If Provider A only has Transfer level access, the PC and CLC communicate to confirm that Provider B is a class of node with whom Provider A may transfer a record. After this confirmation, Provider B will be given a permissions field in the PC, and the proxy re-encryption process begins. Figure 11 depicts the re-encryption process. Steps 9, 10, and 11 in figure 10 correlate to steps 1, 11, and 12, respectively, in figure 11.

To utilize the benefits of both blockchain technology and blinded, distributed re-encryption, the re-encryption process encapsulates multiple transactions be-

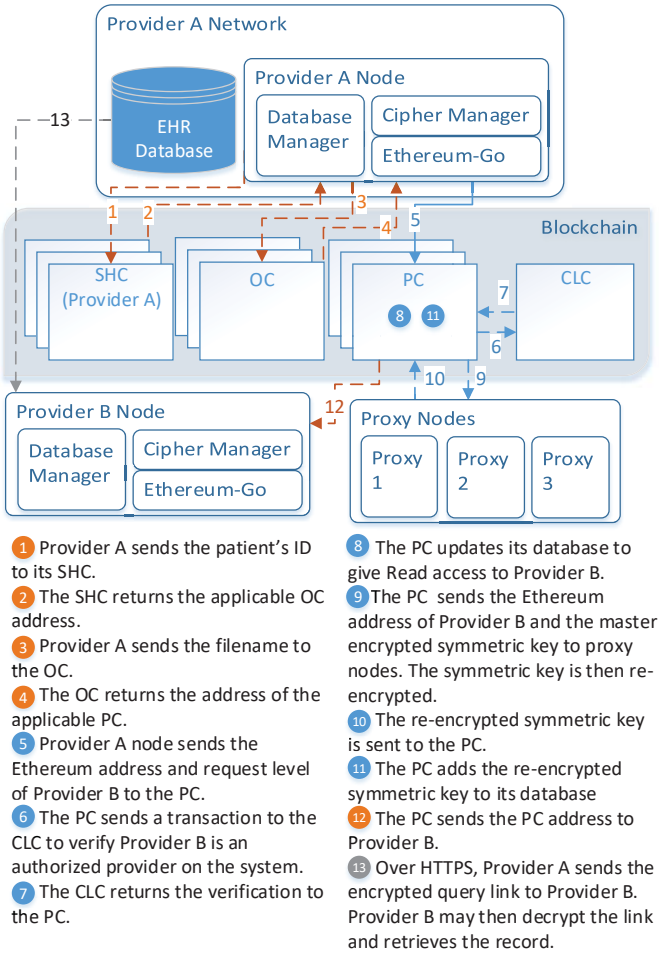


Figure 10: The process for a provider node sending a patient record to another provider. For this process to complete successfully, Provider A must have Transfer level access or higher to the specific record.

tween proxy nodes and an RC. The PC may be programmed to select any RC in the system, making the choice pseudo-random. Moreover, the RC may establish a threshold such that only a certain number of proxies in the group need con-
690 tribute to re-encrypt the key. The RC would be responsible for managing the proxy re-encryption and would then return the newly encrypted symmetric key to the PC. It should be assumed that the chosen proxies know the public key

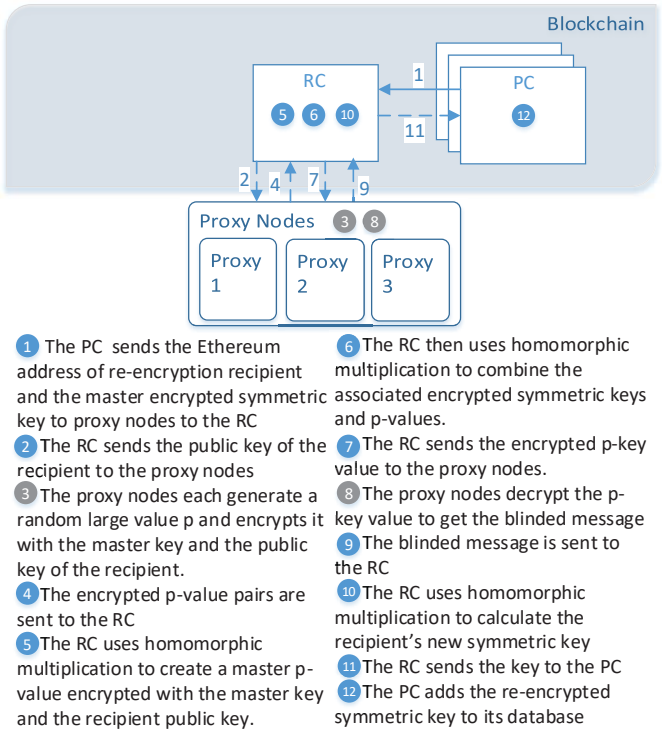


Figure 11: The process for re-encrypting a symmetric key. The proxies used in this process may be pseudo-randomly chosen by the PC, but the proxy group, and thus the RC, must have been established previously.

of Provider B for the re-encryption process. Record transfer is completed when Provider B receives the encrypted query link over HTTPS and the address of the PC from the blockchain. Provider B may then retrieve their key from the PC and decrypt the record.

5. Discussion

5.1. Comparative Performance Analysis

In this section, we perform comparative performance analysis by comparing the estimated computational costs of Ancile and MedRec [12]. We will classify actions into either *off-blockchain* or *on-blockchain* actions. We will use gas

as a unit of measure when discussing on-blockchain actions. As mentioned previously, gas costs are a measure of computational cost in the Ethereum blockchain [9] such that as gas costs go up so does computational time.

MedRec [12] only provides a complete process of adding a record and no specific computational data, so we will specifically be comparing that use case to estimate performance differences. In MedRec, the steps to add a record for a patient involves:

1. A provider sends a request to their EHR manager, a management system that handles local database updates, sending notifications, and provides an interface to view medical records, to add a record.

2. A request is sent to the blockchain to retrieve address of the patient and summary contract from the registrar contract.

3. A request is sent to the blockchain to post a new patient provider relationship contract and link the summary contact with it.

4. Miners verify the requests to the blockchain and once they successfully verify them they are rewarded with bounty specified in the patient provider relationship contract.

5. The summary contract is updated on the patient node and a notification is sent to the patient about the update using EHR manager.

6. Patient then approves or rejects the changes.

7. Depending on the response an appropriate update is sent to the patient provider relationship contract status in the service contract.

8. Patient node sends a signed query request to the providers' database gatekeeper and it checks permissions to see what information can be sent with the query to the patient node.

9. Patient nodes database keeper updates patient nodes local database with the information received from patient node.

Steps 1, 5, 6, 8, and 9 are off-blockchain actions, while steps 2, 3, 4, and 7 are on-blockchain actions. The off-blockchain actions involve hashing links, querying/updating local databases, and sending notifications. Depending on the implementation of the off-blockchain modules, the performance cost can be very low. Now for the on-blockchain actions, since most of them are just retrieving and storing single data value at a time in smart contracts, the gas cost depends on the size of the data value but will still be on lower end of the gas limit compared to doing multiple different actions in one transaction.

In Ancile, the steps to add a record for a patient are detailed in Figure 8. Steps 1, 2, 3, 11, and 12 are off-blockchain actions, while steps 4, 5, 6, 7, 8, 9, and 10 are on-blockchain actions. The off-blockchain actions involve generating query link, hashing it, encrypting it, and database storage/retrieval. Depending on the implementation of the off-blockchain modules, the performance cost can be very low but compared to MedRed, it would be a bit higher since we are encrypting the keys used to decrypt the EMR links. As for the on-blockchain actions, there are quite a few different types of actions include retrieving and storing data values in smart contracts, sending internal transactions to link different contracts, and spawning new contracts using other contracts. The gas cost will depends on the size of the data values being stored and passed but since we also have multiple actions happening in some transaction that include creating more contracts, the gas cost will be half-way to reaching the upper limits of the gas limit. Since Ancile has more steps, especially on-blockchain actions, compared to MedRec, there will be a higher performance cost. But we believe the trade is for the best because we provide a more secure way of allowing providers to store small medical records and links to larger medical records by utilizing symmetric encryption on the keys to decrypt links/data. Since MedRec doesn't provide anymore full processes of some of their use cases, we are limited in direct comparison. However, we can conclude that most of the use cases in Ancile will have higher performance costs and gas costs than MedRec due to the involvement of more modules and steps, but also allow for more features than MedRec.

5.2. Ancile in Context of Healthcare Legislation

The two main bodies of legislation that affect the use of EHRs are the HITECH Act and HIPAA. Both acts are overseen by the United States Department of Health and Human Services (HHS). This research was specifically focused on creating a framework for nationwide EHR management that meets HIPAA requirements. We believe Ancile achieves HIPAA compliance under the Privacy and Security Rules. More specifically, Ancile applies the four technical safeguards listed in the Security Rule.

First, the Security Rule specifies that healthcare infrastructure should have the option to "allow only authorized persons to access electronic protected health information (e-PHI)"[7]. Ancile meets this standard by using identity-checking in PCs and the CLC to confirm that sensitive information is only given to authorized users. The encryption employed by Ancile also prevents unauthorized users from reading PHI even if they managed to access the EHR Database. Moreover, Ancile goes a step further by using the COC to verify legitimate entities prior to registration, controlling who may partake in the permissioned blockchain. This identity validation of providers and third parties in addition to patients is a concept unlike any proposed in related works.

In addition to advanced access controls, Ancile allows for users to "record and examine access and other activity in information systems that contain or use e-PHI" [7]. By nature, a blockchain is meant to keep an immutable record of transactions. Ancile uses both *eth_calls* and transactions to maintain a record of all important actions while still maintaining privacy. This utilization of blockchain technology, specifically the use of Ethereum blockchain protocols and structure, allows for a comprehensive record for auditing. Moreover, Ancile's record-keeping capabilities are also used to meet the technical standard of ensuring "that e-PHI is not improperly altered or destroyed" [7]. The use of hashing the query link and the EHR itself allows users to confirm data integrity.

The last technical safeguard specified by the Security Rule is to "implement technical security measures that guard against unauthorized access to e-PHI that is being transmitted over an electronic network" [7]. In other words, EHRs

ought to be secure when they are sent to entities other than the provider of origin or patient. Proxy re-encryption in Ancile allows for EHRs to be transferred without ever being decrypted. Splitting the re-encryption between multiple nodes and separating the symmetric key, EHR, and query link both further increase the security of the system. Thus, even if an unauthorized user managed to obtain the query link and EHR during the process of transfer, they would not be able to decrypt the information.

By meeting the four technical safeguards, Ancile is compliant with the HIPAA Security Rule. Furthermore, Ancile also meets the HITECH standards for interoperability. HITECH emphasizes that EHR systems "support nationwide electronic exchange and use of information" [3]. By utilizing user-friendly resources like light wallets that may be accessed over HTTPS, Ancile is accessible to all with an Internet connection. Moreover, by placing small records on the blockchain, Ancile makes a shift towards eliminating the need for EHR Databases. This shift may act as a starting point for a system where even providers without large database servers may fully participate.

5.3. Privacy Preservation

Ancile employs several privacy-preserving techniques. By tying patients only to existing account numbers and Ethereum addresses, identifying a specific patient becomes quite difficult. The use of encryption on all sensitive information placed on the blockchain decreases the likelihood of the information within records being accessible by unauthorized actors. Furthermore, by using the SHC, OC, and PC to separate information, there is a heightened level of data obfuscation.

However, while Ancile significantly increases the difficulty of attack, there are still improvements that could be made. First, the strive for the privacy requires

[3] How does the hitech act address barriers to information exchange? URL https://www.healthit.gov/policy-researchers-implementers/ faqs/how-does-hitech-act-address-barriers-information-exchange

a compromise in ease of use. The use of so many different smart contracts, all executing on every full node, means that using the blockchain will not only require a lot of computational power, but will also take time to execute each action. Moreover, because adding a new node to the system and registering a patient relationship are processes that require several steps of verification, a patient may be required to do more during the initial paperwork period at the beginning of an appointment. Further work could investigate ways to simplify the registration process.

Second, by using blockchain technology it becomes impossible, by design, to hide all information. Blockchain analysis could possibly reveal the frequency with which a specific node visits a healthcare provider and which providers they visit. As a result, it would be possible to infer information such as date of birth, medical condition, and general area of residence. While it would be formidable to gather such information and figure out which person is associated with the Ethereum address, it would not be impossible. To remedy this, Ancile could incorporate the use of differential privacy [32], a rigorous privacy model that was created for the express purpose of preserving data privacy while maintaining utility. While technically the data viewable in Ancile remain confidential, a differential privacy scheme would add noise to the transactions in the blockchain, preventing blockchain analysis from inferring extra information. Further investigation could assess the usefulness of differential privacy and how noise might affect the size of the blockchain. Additionally, the possibility for delayed transactions or new mining techniques may also be useful directions for further investigation.

5.4. Scalability

Ancile is founded on several ideas designed to promote scalability. For example, by only storing hashes and small records on the blockchain, much less storage space is needed. Additionally, only few nodes need validate transactions containing data hashes, as part of private transactions. These measures reduce the storage and mining costs of the blockchain as it scales. On the other hand,

the time to search global contracts like the CLC would increase as more users are added to the system. As a future research, we will look for ways to effectively search smart contracts with large local databases or to eliminate the need for global smart contracts.

6. Conclusion

In this paper, we sought to design a blockchain framework for EHR management that could give ownership and final control of EHRs to the patient, securely control who can access documents and track how records are used, allow for secure transfer of records, and minimize ability for unauthorized actors to derive PHI while being HIPPA compliant. The Ancile framework demonstrates a blockchain system that achieves a high-level of decentralization while acknowledging that some nodes ought to be of a higher authority. This study revealed that it would be highly unlikely to completely conceal all information and maintain an accessible and interoperable system, but by using smart contracts to separate information, Ancile still offers significant privacy preservation and data integrity.

Ancile was designed to be implemented over existing systems and it utilizes specific Ethereum tools to create a system that is both cost and storage effective for blockchain technology. The use of encryption and authentication throughout the blockchain demonstrates our prioritization of security and access control. Refusal to use patient data or money as an incentive for blockchain mining, preferring to let use of the blockchain be its own reward to providers and third parties, demonstrates commitment to a patients ownership of their own data. Because of these advantages, the Ancile blockchain would accomplish our original goals to a great extent.

It should be noted, however, that many of the technology used throughout this framework, such smart contracts, and permissioned blockchains, are still in the early stages of their development in the Ethereum community. As a result, the framework rests heavily on their success. We do not propose Ancile as the

remedy to the grander EHR security problem, but we look forward to continued research in the use of blockchain technology to meet legislative standards for medical data and protection of patient privacy.

Acknowledgment

This research was supported by the US National Science Foundation (NSF) under grant CNS 146113.

References

[1] J. Atherton, Development of electronic health records, American Medical Association Journal of Ethics 13 (3) (2011) 186–189.

[2] N. Akpan, Has health care hacking become an epidemic? (March 2016). URL http://www.pbs.org/newshour/updates/has-health-care-hacking-become-an-epidemic/

[3] Breaches affecting 500 or more individuals (June 2017). URL https://ocrportal.hhs.gov/ocr/breach/breach_report.jsf

[4] Securing hospitals: A research study and blueprint. URL https://www.securityevaluators.com/hospitalhack/securing_hospitals.pdf

[5] Health insurance portability and accountability act (2017). URL http://www.dhcs.ca.gov/formsandpubs/laws/hipaa/Pages/1.00WhatisHIPAA.aspx

[6] Summary of the hipaa privacy rule, Web (2013). URL https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html

[7] Summary of the hipaa security rule, Web (2013). URL https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html

[8]

[9] G. Wood, Ethereum: A secure decentralized transaction ledger (2014).
URL http://gavwood.com/paper.pdf

[10] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008).

[11] V. Buterin, On public and private blockchains, Ethereum Blog (2015).

[12] A. Ekblaw, A. Azaria, J. D. Halamka, A. Lippman, A case study for blockchain in healthcare: "medrec" prototype for electronic health records and medical research data, in: Proceedings of IEEE Open & Big Data Conference, 2016.

[13] L. A. Linn, M. B. Koo, Blockchain for health data and its potential use in health it and health care related research (2016).

[14] Quorum whitepaper.
URL https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf

[15] Electronic health records implementation with blockchain, bpm, ecm, and platform.
URL http://improving-bpm-systems.blogspot.ch/2016/07/electronic-health-records-ehr.html

[16] D. Ivan, Moving toward a blockchain-based method for the secure storage of patient records (2016).

[17] C. Brodersen, B. Kalis, C. Leong, E. Mitchell, E. Pupo, A. Truscott, Blockchain: Securing a new health interoperability experience (2016).

[18] K. Culver, Blockchain technologies: A whitepaper discussing how the claims process can be improved (2017).
URL https://www.healthit.gov/sites/default/files/3-47-whitepaperblockchainforclaims_v10.pdf

[19] K. Peterson, R. Deeduvanu, P. Kanjamala, K. Boles, A blockchain-based approach to health information exchange networks (2016).

[20] C. Cachin, Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[21] R. G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: An introduction, R3 CEV (2016).

[22] R. C. Merkle, A digital signature based on a conventional encryption function, in: Conference on the Theory and Application of Cryptographic Techniques, 1987, pp. 369–378.

[23] Quorumchain consensus (2017).
URL https://github.com/jpmorganchase/quorum/wiki/QuorumChain-Consensus

[24] G. Wood, Ethereum: A secure decentralized generalised transaction ledger eip-150 revision (2014).

[25] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, ACM Transactions on Information and System Security (TISSEC) 9 (1) (2006) 1–30.

[26] L. Zhou, M. A. Marsh, F. B. Schneider, A. Redz, Distributed blinding for distributed elgamal re-encryption, in: IEEE International Conference on Distributed Computing Systems (ICDCS), 2005, pp. 824–824.

[27] Go ethereum (2017).
URL https://github.com/ethereum/go-ethereum/blob/master/README.md

[28] V. Buterin, State tree pruning (2015).
URL https://blog.ethereum.org/2015/06/26/state-tree-pruning/

[29] P. Szilagyi, Eth/63 fast synchroniation algorithm (2015).
URL https://github.com/ethereum/go-ethereum/pull/1889

[30] Step-by-step guide: Getting started with ethereum mist wallet (2016).

[31] P. McCorry, S. F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy., IACR Cryptology ePrint Archive 2017 (2017) 110.

[32] C. Dwork, Differential privacy, in: Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), 2006, pp. 1–12.