Towards Secure Interoperability between Heterogeneous Blockchains using Smart Contracts

Gaby G. Dagher
CS, Boise State University
Boise, Idaho, USA
gabydagher@boisestate.edu

Chandra L. Adhikari
CS, Boise State University
Boise, Idaho, USA
chandraadhikari@u.boisestate.edu

Tyler Enderson
CS, Boise State University
Boise, Idaho, USA
tylerenderson@u.boisestate.edu

***Abstract*—Achieving data confidentiality and privacy while maintaining secure access is essential in various fields, including in the medical sector. Implementing a blockchain-based technology to secure sensitive data ensures that the users own their data and have control over who can access it. While blockchain technology is still in its infancy, it is the cutting-edge of research in many industries and institutions. The decentralized nature of blockchain technology and the presence of smart contracts in Ethereum are two major features that can be utilized to create a novel data sharing and access system that is secure, flexible, and more reliable. In this paper, we investigate the use of smart contracts between heterogeneous blockchains for the purpose of achieving secure interoperability for data sharing and access control. As a proof of concept, we propose and implement a record management system for healthcare data, where access to healthcare providers' databases is managed through a private blockchain, only available to healthcare providers, and patients access their medical records through a public blockchain. Additionally, we develop a set of smart contracts for each blockchain to control access, manage storage, and enable interoperability between the two blockchains.**

***Keywords*—Blockchain; Ethereum; smart contract**

I. Introduction

The modern world is moving rapidly with technology, digitalizing record management as well as utilizing newer, data-driven equipment. Data assists in further advancing technology in many ways, particularly aiding development of scientific research, business practices and government policies. Collecting, storing and accessing data is a common practice nowadays across industries, institutions, and governmental organizations. However, to provide the most benefit, data needs to be shared across organizational boundaries, enabling authorized users to access the data while protecting sensitive information. The storage of data containing private and sensitive information while enabling availability is a challenging and difficult task that requires continuously maintaining an interoperable sharing system that allows access in a secure manner. In fact, one of the major issues with existing systems is insufficient security controls for users accessing or sharing files [1], as there have been major concerns about user privacy and the limitations of controlling access to private data [2]. For example, in the past couple of years, more medical records breaches than ever before have been recorded. In 2015, there was 270 security breaches hitting 113,267,174 records, while in 2016, 329 breaches occurred that involved 16,471,765 records [3].

In recent years, blockchain has emerged as a potential technology for solving current security, privacy, and interoperability challenges in several domains, including financial services [4], Internet of things (IoT) [5], [6], and healthcare [7]. Blockchain, further described in Section II, maintains a distributed, immutable ledger that contains a history of what has occurred on the blockchain. Immutability allows records to persist for the life of the blockchain and can aid data integrity over long periods of time. Additionally, many proposals for using blockchain incorporate access control measures that benefit from decentralized authority without risking the privacy of data [2]. Not only can blockchain technology manage access control, but also the storage of files on off-chain databases. Advantages of blockchain technology can be incorporated through the use of Ethereum [8], a prominent implementation platform for building blockchain-based systems. Ethereum provides high utility through Turing-complete smart contracts and shortened block intervals. While there have been advances in using single-blockchain systems in different domains, there has been limited work on analysing how multiple-blockchains can be utilized to build systems that demonstrate interoperable management of smart contracts and data.

In this paper, we investigate the use of smart contracts between heterogeneous blockchains for the purpose of achieving secure interoperability for data sharing and access control. According to [9], there is a need for records management system that is flexible enough to access files universally, protect privacy, provide a complete control over individual's information, and enables patients to determine which records to be visible. Therefore, as a proof of concept, we propose and implement a record management system for healthcare data, where access to healthcare providers' databases is managed through a private blockchain (only available to healthcare providers), and patients access their medical records through a public blockchain. Data stored directly on the blockchain, such as links to files, is encrypted so that only authorized users are able to access it. Additionally, we develop smart contracts for each blockchain, to control access and manage storage. Each smart contract can autonomously execute other smart contracts on interoperable public and private blockchains to complete tasks. Incorporating this type of records management contributes to the strength of system security so that only the approved parties have permission to view the data.

The rest of the paper is organized as follows: Section II introduces different aspects of blockchain technology. Section III reviews the related literature. Section IV describes the concept of secure file access between heterogeneous blockchains using smart contracts and Ethereum's blockchain technology. Sec-

tion V is the discussion and finally, we conclude the paper in Section VI.

II. Preliminary

In this section, we provide background information on blockchain technology that contributes to our solution.

A. Blockchain Technology

1) Introduction: Blockchain, first introduced in Bitcoin [10], is a decentralized ledger of verified transactions across a peer-to-peer network. The Genesis block, an initial block without any transactions, is the predefined start of the blockchain; subsequent blocks following the genesis block consist of blocks constructed with transactions that through a consensus protocol, have been verified by special nodes in the network called miners. Nodes on the network keep an updated version of blocks that continues to grow. Blocks of transactions confirmed by miners are appended to the end of the chain as new transactions appear on the network. These transactions are created when a user sends/calls another user with optional input data, or through execution of a smart contract, and reside in the transaction pool of the network where all of pending transactions are stored. Pending transactions are pulled from the pool by miner nodes who verify them in order to create blocks. Once the block is mined and added to the chain, it becomes part of the append only chronological ledger. By initial design, the blockchain technology is used in the fashion of a financial ledger, but we have extended it to support secure file access.

2) Consensus Mechanisms: The consensus protocol adopted by a blockchain determines which block to append and how to do so, confirming validity of contents and block construction in the process. Parameters within the consensus protocol can be changed to modify block characteristics like structure and intervals between blocks. Bitcoin [10] and Ethereum [8] uses a Proof of Work (PoW) algorithm that requires miner nodes to invest computational power to add security and immutability to the blockchain. PoW has been criticized for inefficient use of electricity, computational power and hardware, and resulting in the introduction of alternatives such as Proof of Stake (PoS) or Proof of Retrievability (PoR). In PoS, instead of *all* miner nodes competing to create the next block like PoW, a miner node is chosen to invest computational power through pseudorandom methods weighted in relation to degree of ownership, or credibility, or reputation, etc. This type of miner selection results in far less resource expenditure. PoR similarly conserves resources through use of invested storage by miner nodes and can support blockchain implementations by functioning as a database.

3) Blockchain Types: Blockchains can be categorized as *public*, *consortium*, and *private*.

Public blockchain. A public blockchain is a chain that anyone in the world can read, send a transaction to a valid user, and can involve the consensus process. The consensus process is a process to determine what block gets appended to the end of the chain. Homogeneous or heterogeneous blockchains can have various types of consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), Proof of Retrievability (PoR), and so on. The blockchain in general is considered s decentralized network where there is no central authority controlling the system. Blockchain is the underlying fabric for cryptocurrency system scan involve the consensus process and has a design pattern consisting of three main components: a distributed network, a shared ledger and digital transactions [1]. A distributed network simply means that all the participating nodes on the network store the unaltered copy of the blockchain and assist in certifying the digital transaction. A shared ledger is a ledger where all confirmed transactions will be stored after the majority of the participants agree on each transaction. Once stored, it will exist in all participating nodes and cannot be altered. Lastly, a digital transaction contains digital proof of validation such as a transaction hash, associated block, and other necessary input data. Additionally, a transaction can also carry extra data in the hexadecimal representation serving as an argument to another transaction [11]. The transaction information is encrypted and has been signed digitally to guarantee data integrity. Confirmed transactions are then committed to a block, which contains a cryptographic hash of the previous block. Each block is linked with its previous block in a linear order, forming a long blockchain.

Consortium (Permissioned) Blockchain. A consortium blockchain is a controlled blockchain where pre-define nodes are allowed to perform certain tasks. It is also considered a semi or partial decentralized network due to the fact that defined nodes can control the network by altering the transactions. The consensus process in such blockchains can be customized, where certain nodes in the network can also be a part of the consensus team. The consensus algorithm could be modified to use different ideas such as voting based concept. The consensus policy can be set where a certain number of nodes in the network can vote or sign the block before committing the block to the chain [12]. Another feature of the consortium blockchain is that a transaction can be set to read-only mode for the public, but transaction creation can be kept private so that no other parties can send a transaction to the network.

Private Blockchain. A private blockchain is generally similar in features with consortium blockchain. It is also considered as a centralized blockchain where a central authority has permission to control the transactions in the entire chain from adding, deleting, or modifying the data in any transaction [12]. Similar to consortium blockchains, read access can be defined to specific participants along with limitations on creating transactions. The private blockchain is mainly used to manipulate databases or other private applications to be used for sensitive data. In order for a transaction to be valid, the pre-defined list of nodes must verify the transactions. While the verification is only happening inside the private network, it is assumed that those nodes have a high level of trust. Due to the nature of private blockchains, not all nodes are required to verify transactions. This will not only increase the transaction speed in the private blockchain, making it the fastest blockchain-based solution, but it will also reduce the amount of work [13].

B. Smart Contracts

In Ethereum [8], a smart contract is a piece of code consisting of a set of functions that gets executed using Ethereum's execution environment. Each contract has its own storage that can only be modify by its contract. The modified storage of a contract is considered the state of the system. For example, if a setter function is implemented in the code, then it can be used multiple times by sending a transaction with the new value to modify the old value of a static variable. This modification is considered a change in the contract's storage.

Solidity [14] is a statically typed high-level language that is being used to develop smart contracts. The source code is compiled down to Ethereum Virtual Machine (EVM) byte code. Contracts are mainly used to automate certain tasks and operate like autonomous agents. Similar to autonomous vehicles, contracts can be coded to follow a set of rules or policies and enforce actions such as logging messages or data collection automatically based on the defined policy. Once a contract is created and deployed, its source code cannot be removed from the blockchain even if it kills itself. The compiled version of the source code is permanently stored within the transaction that creates the contract. Successfully deployed contract have their own address just like any other users in the network. The process of killing contracts is also called *contract suicide*, which takes place by sending a kill command to itself. Upon killing itself, contract's address will no longer be accessible and cannot receive any further transactions. Sending a transaction to a contract's address is the process of triggering a contract causing the execution of the code inside the function that is being called with the specified parameter. Additionally, a contract is capable of creating another new contract by sending a transaction [15]. Fig. 5 demonstrates an example of a smart contract written in Solidity.

III. Related Work

The blockchain is an emerging technology and many concepts are being researched that expand how blockchain is used. In this section, we review research that relates to access control, storage, and interoperability between separate blockchains.

A. Sidechain, Software Connector and Interoperability

Enabling interoperability between multiple blockchains through sidechains is one area of research expanding on possible implementations. Pegged sidechains, which allow for the transferring of assets in both directions directly between chains, i.e. tokens or cryptocurrency, without the need for an exchange are covered in detail by [17]. The sidechain receives assets from the parent chain through transactions sent to a special address, locking the assets on the parent chain to enable use on the sidechain. Pegged sidechains are then able to utilize the assets with a modified version of the parent blockchain protocol and can freely transfer the assets back to the parent. This type of blockchain interoperability is limited; only assets are transferred in a 1-to-1 relationship with no increase in the amount of total assets.

The authors in [15] examine the suitability of blockchain technology to perform as a software connector, an interaction mechanism between different components such as pipes or sockets. Further, design decisions for blockchain specific components on-chain and off-chain are identified, impacting the usability for requirements such as access control or data storage.

B. Blockchain as Access Control

Blockchain has seen numerous proposals that incorporate various access control methods through transactions or smart contracts [7], [2], [15] that are modeled on or interact with traditional access control systems. As a mechanism to employ access control, blockchain is also interestingly portrayed as a solution to authentication for IOT use [5], [6]. Typically in proposals where blockchain is used for access control, transactions or smart contracts are used to pass an obfuscated symmetric key between a sender and recipient.

C. Privacy and Data Storage

As increasing amounts of personal and private data are available through the connected world there is a need to create storage systems which maintain privacy [2]. While there are methods for returning queries on data that maintain privacy, anonymization techniques to publish data while protecting private information through for example*k-anonymity* or *t-diversity*, the use of blockchain to store data can further protect privacy by eliminating a third-party, centralized trusted authority that could fail.

MedRec [7] is a project, distributed by MIT Media Lab, involving public blockchain and multiple smart contracts to protect and manage electronic health records (EHR). The blockchain solution is proposed to replace how EHR are currently managed by improving the ability for transfer and maintenance of records between providers and patients. Multiple smart contracts present in the system have a specific purpose or task in determining data access permissions and the interaction between the contracts establishes secure communication for management of records which protects privacy.

Quorum [16] is an open-source project organized by JP Morgan Chase bank. Quorum enables creation of private and or permissioned blockchains based on the Go implementation of the Ethereum Protocol which implement further privacy controls. Through implementation of a tag to serve as an identifier for private transactions, sensitive information is segmented so that it is only available to parties involved in the transaction [16]. Private smart contracts are encrypted on the public blockchain, and are stored only with parties involved in the corresponding transaction. The blocks of transactions in the QuorumChain are certified based on voting consensus algorithm and in Quorum, a private transaction creates a private contract and the state is represented in its own patricia-merkle trie. Interoperability can be limited however as creation of a private contract using a public transaction is prohibited due to the state of private and public transactions being recorded is in two different patricia-merkle trie.

The proposed solution in this paper advances on these research areas and is maintained by deployed contracts on public and private blockchains to perform autonomous tasks such as adding a link to a file, registering a patient, or changing the file read permission. We summarize the comparative information of the closely related work, alongside our proposal, in Table I.

TABLE I. COMPARATIVE EVALUATION OF MAIN FEATURES IN EACH CLOSELY RELATED WORK (PROPERTIES IN COLUMNS ARE POSITIONED AS AVAILABLE FEATURE DENOTED BY ●)

Related Work	Types of Blockchain			Security			Interoperability
	Public	Private	Consortium	Access Control	Privacy	Storage	
MIT Media Lab - MedRec (PSD) [7]	●			●		●	
JP Morgan Bank - Quorum (SED) [16]	●	●	●	●	●	●	
Pegged sidechain [17]	●	●	●				●
IoT [5][6]	●			●			
Our Proposal: Section IV	●	●	●	●		●	●

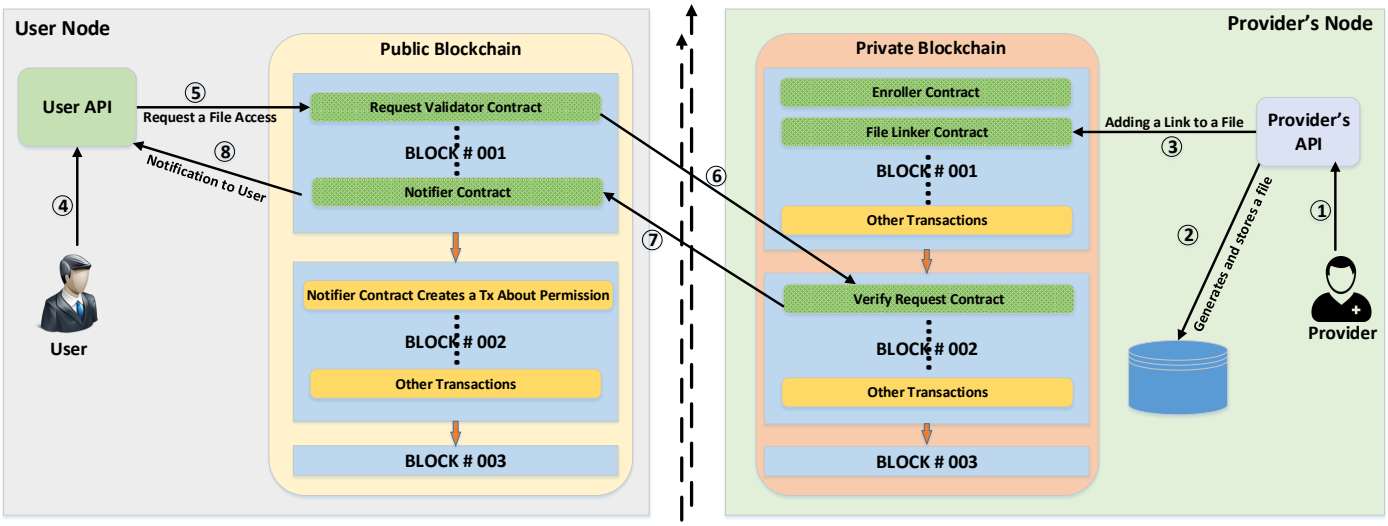

Fig. 1. Overview of the proposed system in chronological Order. All rectangles in green color with pattern indicate smart contracts. The orange rectangles indicate transactions and each blue box represents a block that is linked to its previous block inside the chain. The light yellow box represents public blockchain and the pink box represents the private blockchain. The light grey box represents the user's (patient) node and light green box indicates the provider's (doctor) node.

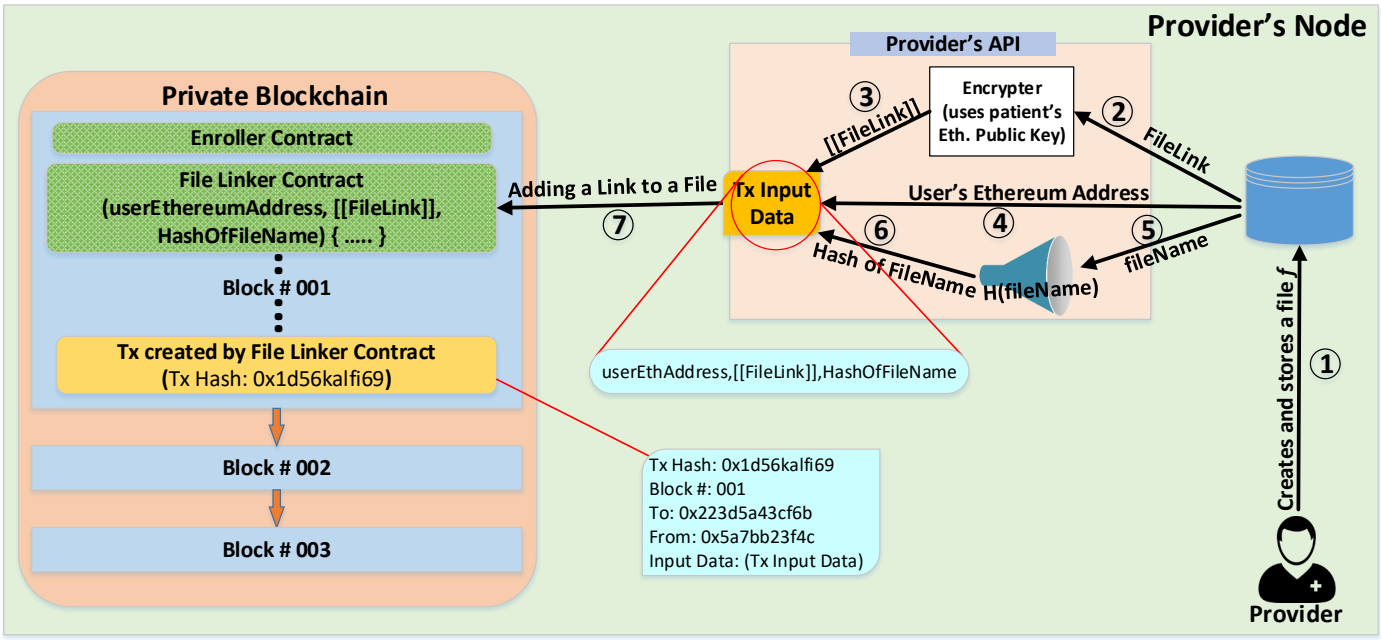

Fig. 2. The process for registering a file on the private blockchain by a provider. It starts with creating a new file and storing it in the local database. Steps 2, 3, 4, 5, 6, and 7 prepare the input data to the *File Linker* smart contract. Once the *File Linker* is called with the input data, it will create a record in the form of transaction on the private blockchain indicating that the specified file link is mapped to the specified patient's Ethereum address. This record will be searchable when verifying requests from patients.

IV. Proposed System

In this paper, we explore how Ethereum's blockchain technology can be utilized to implement protocols for secure file access using smart contracts. Our solution consists of smart contracts that pass sensitive input data from public to private blockchains and vice versa. We incorporate a private blockchain to prevent unauthorized users from having access to the private network. In order to achieve security, we create smart contracts, e.g. *Verify Contract* (Fig. 5), that runs a multi-verification process before allowing access to a specified file. Transactions hold the same structure as the standard Ethereum's transaction, with the input data field being required instead of being optional. This input data field is used to pass various encrypted messages and sensitive information. The mining process for all of our transactions is adopted from Ethereum's standard process.

A. Solution Overview

Our design and its structure uses Ethereum's public and private blockchains, smart contracts, and a local database to provide patients with secure access to their electronic health records. The proposed approach allows a patient to send requests to its provider's system through the blockchain. Smart contracts on the Ethereum blockchain are utilized to automate tasks and achieve access control. Consider a simple case where a patient is trying to access a file from his medical records. The process flow of this scenario is shown in Fig. 1. We assume that both the provider and the patient are registered in this system. The initial step begins with the provider creating a new report and then inserting it into a local file repository (local database). For the file to be linked, the *File Linker* smart contract will be called with a structured parameter input. Each link is generated uniquely for a single file. After successful *File Linker* execution, a transaction with the input data will be created. When a patient later requests that document, the *Request Validator* smart contract is called, which then calls a *Verify Request* smart contract that resides in the private blockchain. The request is verified by *Verify Request* contract using an algorithm which performs three-step verification process. Once validated, a *Notifier* contract on the public blockchain is called that passes an encrypted message to the requester. This encrypted message contains the actual link to the file. A transaction with input data will be created upon successful execution of this contract. Once the transaction arrives on the public blockchain, the user will be notified through its Ethereum wallet, which will then decrypt the encrypted link to obtain the actual link to the file.

B. File Registration in Private Blockchain

Before requesting to view a document, the document must be registered on the private blockchain via a transaction. When creating this transaction, the provider is required to specify the default permission along with the actual file. An Ethereum address is used to identify the initial permission. For example, if a new medical file is generated for patient P, then the provider will utilize P's Ethereum address to set valid permission for P to view that file. All sensitive information such as file name and download link are encrypted before passing them as input data to the *File Linker* smart contract as shown in Fig. 2.

The local database shown in Fig. 2 belongs to the provider's node where *file name* f, *file link* l, and the *Ethereum address* of the patient are saved. The provider's interface is responsible for using a public-key cryptosystem (e.g. Elgamal [18]) to generate encrypted file link $[\![l]\!]$ that maps to file f.

The provider's interface captures the patient's public key and uses it to generate ciphertext $[\![l]\!]$. Additionally, the file name f is hashed using a one-way hash function (i.e. SHA-256) to generate 256-bit (32-byte) hash. Reports that belong to a patient are determined by using his/her Ethereum public address, which is a part of the transaction input data. These pieces of text are formatted into a single piece of data. The format follows this outline: $\langle$userEthAddress$\rangle\langle[\![FileLink]\!]\rangle\langle$hashOfFileName$\rangle$, where *userEthAddress* is the Ethereum address of the patient, $[\![FileLink]\!]$ is the secure link to file f, and $\langle$*hashOfFileName*$\rangle$ is the hash the file name f. The input data is passed into the *File Linker* smart contract resulting in a transaction with the provided input data. This process of injecting a transaction with encrypted input is considered as creating a record that is searchable by patient requests.

C. Patient's Access Request to a File

Patient's request to view his/her file securely is a key feature in our proposed system. By default, we assume that the patient knows the name of the file being requested. In order to verify that the request is from a an authorized and registered patient, the patient generates a hash of the file name and then uses its private key to sign the hash. The signature and the hash of the file name are merged together to create the input data for the *Request Validator* smart contract. Through its Ethereum wallet, the patient calls the *Request Validator* to generates a transaction that which will be placed in the public blockchain as a record of request. As shown in Fig. 3, the *Request Validator* contract calls the *Verify Request* contract in the private blockchain and passes to it the input data. This call will result in a transaction in each of the private and public blockchains. Verifying the signature and the hash of the file name happens inside a smart contract.

Once all of these steps are verified, the *Verify Request* contract will send an encrypted message to the *Notifier* smart contract creating a transaction in the public and private blockchains. *Notifier* contract will then execute and create a transaction involving patient's Ethereum address and the message that was passed by *Verify Request* contract. This passed message contains the secure link to access the file encrypted using Elgamal cryptosystem and the public key of the patient. The patient's user interface will receive a notification from the *Notifier* and inform the user about the permission status of the request.

D. Smart Contract Structure

Fig. 5 illustrates how all contracts collaborate. In order to achieve interoperability, this project uses a total of five smart contracts: three resides in the private blockchain and the remaining two reside in the public blockchain. The private blockchain has the *Enroller*, *Request Validator*, and *File Linker* contracts, whereas the public blockchain has the *Verify Request* and *Notifier* contracts.

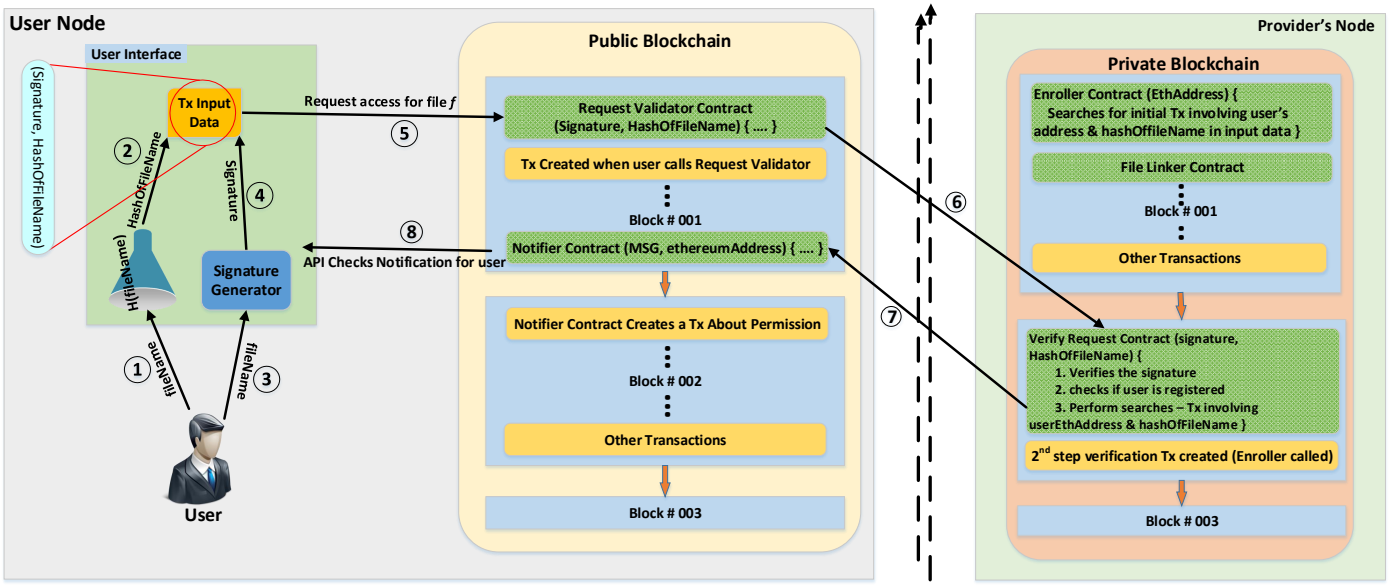

Fig. 3. This figure shows the flow in chronological order of a user requesting access to a file in the system. At this stage, the file should already be deposited (registered) in the private blockchain. Firstly, the patient will use his private key to generate a signature of the hash of file name. Upon finishing Steps 1 through 4, the input data to the *Request Validator* will be created. Once the *Request Validator* calls the *Verify Request* contract, Step 3 will start resulting in an encrypted message being sent to the *Notifier* contract. Regardless of whether the request is valid or not, the message will be sent by the *Notifier* contract to the user's Ethereum address. Finally, the user's API is responsible for consistently checking for a new message and notifies the user about the request result.

1) Enroller Contract: This registrar contract is the initial contract that gets executed when a provider is registering a new patient into their system. Each call to *Enroller* contract will issue a transaction in the private blockchain making it visible to the members of the private blockchain. This transaction includes encrypted patient's sensitive information in the input data section, which enables efficient search if needed. Since the transaction will reside in the private blockchain, it is more secure than public blockchain - preventing public viewing of private transaction detail information (see Fig. 4).

2) File Linker Contract: The purpose of this contract is to make sure that a new secure link is mapped to specified file. The input to this contract can be represented as P_1, P_2, and P_3, where P_1 refers to the patient's Ethereum address, P_2 represents an encrypted text of the link, and P_3 is the hash of the file name to which will be linked. Upon successful execution of this contract, a transaction will be issued to the private blockchain containing P_1, P_2, and P_3. We use this information to search through the list of transactions to verify that a valid user is requesting the access. Each secure link will be used per file when a provider is linking the file.

3) Request Validator Contract: This is a global contract that gets executed in the public blockchain. All user executing this contract needs to be a node in Ethereum's public network. *Verify Request* contract is responsible for securely passing the input data to another contract that resides in the private network of our system. The securely passed input data holds the format as follow: ⟨signature⟩⟨hashOfFile⟩. The purpose of signature verification is to authenticate the patient who is sending the request. The hash of the file will be used to search through the transactions in the private blockchain.

4) Verify Request Contract: This is another contract that resides in the private blockchain for strengthening the security of the system. This contract is responsible for verifying all inputs and determines the validity of a request by checking the permission on the requested file. It takes two parameters: the signature and the hash of the file name that are passed by the patient and then employs an algorithm that uses a three-step verification procedure:

1) *Signature Validation*: The signature passed is verified using the patient's public key and the hash of the file name to confirm that it was in fact sent by that specific patient.
2) *Verification of Registration*: Upon successful verification of the signature, the *Verify Request* contract calls the *Enroller contract*, which will verify whether an address is registered in the private network.
3) *Transaction Search*: This step involves searching for a transaction in the private blockchain by scanning the input data section for an Ethereum address and file hash match. Initially, when a provider inserts a file into the system, permission is given to view the document for specific patient based on his/her Ethereum address. If there is no transaction that contains the hash of the file name that the patient is looking for along with their address, then it simply means that there is no document in the system. Hence, this request will fail. However, if there is a transaction that has the hash of the file name along with the right address of the patient, then the *Verify Request* contract will return the encrypted secure link. Once the transaction is found, it is considered as a valid request and the encrypted secure link of that file will be sent to the *Notifier* contract residing in the public blockchain as a parameter. Fig. 5 is the code for the smart contract that verifies patient's requests implemented in Solidity language [14].

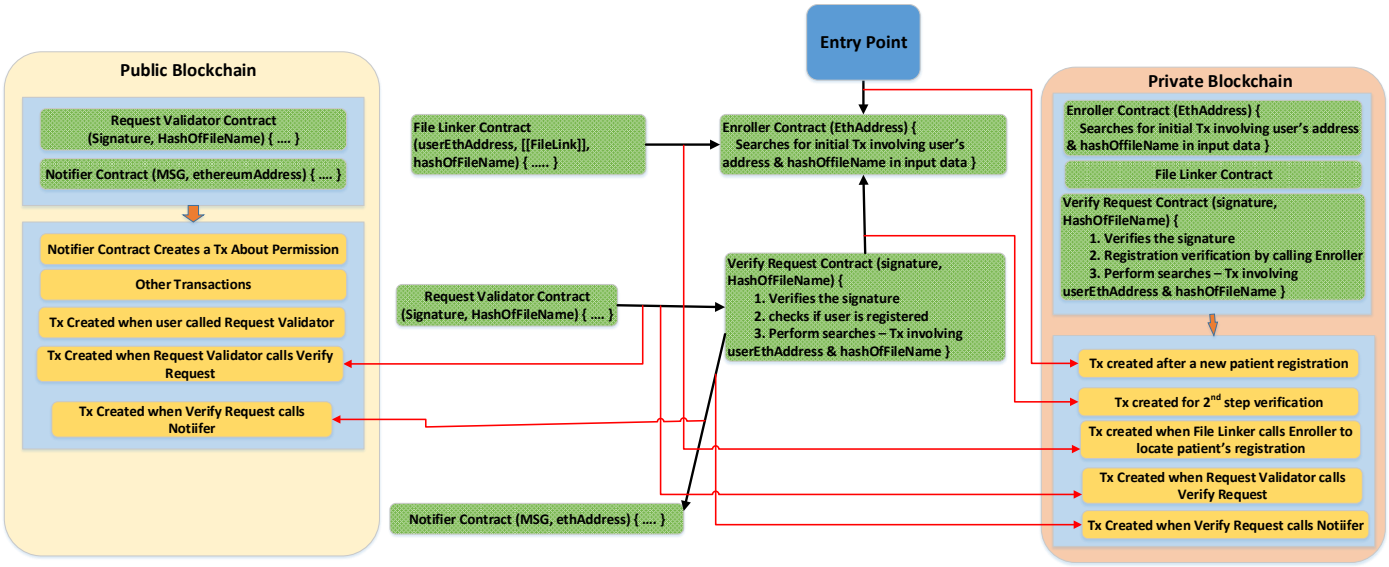

Fig. 4. Interaction between all of the contracts in our system. All of the black arrows going into the contracts indicate that it is being called by another contract located at the starting point of the arrow. All of the red connectors are representing a transaction happening due to the call between two contracts. Each red arrow is either going into the public or the private blockchain, which indicates that the transaction will reside on the specified blockchain. There is a total of two transactions going into public blockchain due to the call between shown contracts and five other transactions are going into private blockchain.

5) Notifier Contract: This contract is deployed in the public blockchain with the purpose of passing a message between two parties. It takes the receiver's address and the message as parameters. When any contract passes a message along with the destination address, this contract will perform its task by creating a transaction with input data on the public blockchain. In such case, the requester must be using an interface that checks the public blockchain consistently and notifies the patient when a transaction arrives that involves the patient. In our system, the actual message that gets posted on the public blockchain is encrypted and is stored as transaction input data. As shown in Fig. 3, the *Notifier* contract takes the encrypted message from *Request Validator* contract and creates a transaction passing the message as input data. This encrypted message will be visible to anyone, but only the patient who has the appropriate private key can decrypt it and obtain the file link. After the patient opens the notification, he/she will be able to use his interface to decrypt the link and to follow the decrypted link to access the file.

V. Discussion

In this paper, two heterogeneous blockchains are used: *private* and *public* to provide interoperable and secure data sharing and access control. These blockchains, however, are both Ethereum blockchains. Our proposed solution does not support a hybrid system, such as a Bitcoin [10] blockchain and an Ethereum [8] blockchain, since these platforms employ different protocols, and Bitcoin does not supported smart contracts. The requirement for using blockchains of the same type limits the application of our proposal between existing, non-compatible blockchains. Furthermore, our system stores data in the provider's local database, which is a separate layer outside the blockchain. We assume that this local database stores the file name, a secure link, the patient's Ethereum address, and the public key of the patient. The database will only be used by the provider when adding a new file and when retrieving the secure link upon valid request from the patient.

VI. Conclusion and Future Work

In this paper, we demonstrate how Ethereum's private and public blockchains, along with smart contracts, can be used to implement a secure data sharing and file access system. Data privacy is achieved by encrypting, hashing, and performing a three-step verification process so that no unauthorized user can access sensitive data of patients. The proposed system enables interoperability between different healthcare providers by enabling their systems to communicate, exchange data, and use the information that has been exchanged.

For future work, our goal is to improve the proposed system to make it compatible with HIPAA [19] rules with respect to data sharing. Moreover, to increase robustness and adaptability of our proposed solution, we will investigate how to get rid of the extra data layer, i.e. provider's database, and have the data records stored directly on the private blockchain.

References

[1] L. A. Linn and M. B. Koo, "Blockchain for health data and its potential use in health it and health care related research."

[2] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 180–184.

[3] Largest healthcare data breaches of 2016. [Online]. Available: http://www.hipaajournal.com/largest-healthcare-data-breaches-of-2016-8631/

[4] M. Swan, *Blockchain: Blueprint for a new economy*. "O'Reilly Media, Inc.", 2015.

[5] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[6] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, "World of empowered iot users," in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, April 2016, pp. 13–24.

[7] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "A case study for blockchain in healthcare:medrec prototype for electronic health records and medical research data," 2016.

[8] G. Wood, "Ethereum: A secure decentralized transaction ledger," 2014. [Online]. Available: http://gavwood.com/paper.pdf

[9] K. D. Mandl, D. Markwell, R. MacDonald, P. Szolovits, and I. S.

VerifyContract.sol

```
pragma solidity ^0.4.0;
contract VerifyRequest {

address public patient_add;
address public notifier;
address private enroller;
string private patient_public_Key;
string private fileLink;

    //Creates a new verify request
    function VerifyRequest(string sig, string hashOfFileName, string public_key) {
        patient_add = msg.sender;
        patient_public_Key = public_key;
        // The secure link to a file.
        fileLink = "www.tinyurl.com/fileLink12u76d";
        notifier = 0xd6Be85dAd78F36f12135a001aD09B5Ee5179d8D0;
        enroller = 0xe5Be85dAd358F39f12135a00DD09B5Ee51794457;
        //First step: sig verification
        int retValue = Verifysig(sig, public_key);
        if(retValue != 0){
            notifier("Bad sig was provided.");
        } else {
            // Second step: Check for registration
            int retV = VerifyEnrollment(patient_add);
            if(retV != 1) {
                notifier("Patient Not Registered.");
            } else {
                // Third step: Check for Tx involving address
                bool retVal = VerifyTxExistance(patient_add, hashOfFileName);
                if(!retVal){
                    notifier("Permission denied.");
                } else {
                    notifier("Valid Request. File link is:" + fileLink);
                }
            }
        }
    }
    // Verification of the signature
    function Verifysig(string sig, string public_key) returns (int retVal) {
        // returns 0 if verified successfully
        retVal = algorithmToVerifysig(sig, public_key);
        return retVal;
    }

    // Verify the registration and search for initial Tx
    function VerifyEnrollment(address patient_add) returns (int retVal) {
        // returns 1 if verified successfully
        retVal = Tx_search_algorithm(patient_add);
        return retVal;
    }

    // Verifies that a transaction exist on private blockchain with the hash
    function VerifyTxExistance(address patient_add, string hashOfFileName) returns
    (bool) {
        string temp_hash;
        string output = hashSearch(patient_add, hashOfFileName);
        address outputA = addressSearch(patient_add, hashOfFileName);
        if((output != hashOfFileName) && (patient_add != outputA)){
            return false;
        } else {
            return true;
        }
    }
}
```

Fig. 5. *Verify Request* smart contract code implemented in Solidity [14].

Kohane, "Public standards and patients' control: how to keep electronic medical records accessible but privatemedical information: access and privacydoctrines for developing electronic medical recordsdesirable characteristics of electronic medical recordschallenges and limitations for electronic medical recordsconclusionscommentary: Open approaches to electronic patient recordscommentary: A patient's viewpoint," *Bmj*, vol. 322, no. 7281, pp. 283–287, 2001.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electionic cash system," Unpublished, 2008.

[11] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Zanella-Béguelin, "Short paper: Formal verification of smart contracts."

[12] J. G. BitFury Group, "Public versus private blockchains: Part 1, permissioned blockchains."

[13] Private versus public blockchains: Is there room for both to prevail? [Online]. Available: https://medium.com/@Magnr/private-versus-public-blockchains-is-there-room-for-both-to-prevail-b97040dacfb

[14] Solidity. [Online]. Available: https://solidity.readthedocs.io/en/develop/

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

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

[17] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.

[18] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*, 1985, pp. 10–18.

[19] Health insurance portability and accountability act (hipaa). [Online]. Available: https://www.hhs.gov/hipaa/