Toward Secure, Privacy-Preserving, and Interoperable Medical Data Sharing via Blockchain

Hao Jin[a], Chen Xu[a], Yan Luo [a], Peilong Li[b], Yu Cao[a] and Jomol Mathew[c]

[a]*Electrical & Computer Engineering Department, University of Massachusetts Lowell*
[b]*Department of Computer Science, Elizabethtown College*
[c]*Department of Quantitative Health Sciences and Information Technology, University of Massachusetts Medical School*

***Abstract*—In the era of cloud computing and big data analysis, how to efficiently share and utilize medical information scattered across various care providers has become a critical problem. This paper proposes a new framework for sharing medical data in a secure and privacy-preserving way. This framework holistically integrates multi-authority attribute based encryption, blockchain and smart contract, as well as software defined networking to define and enforce sharing policies. Specifically in our framework, patients' medical records are encrypted and stored in hospital databases, where strict access controls are enforced with attribute based encryption coupled with privacy level classification. Our framework leverages blockchain technology to connect scattered private databases from participating hospitals for efficient and secure data provision, smart contracts to enable the business logic of clinical data usage, and software defined networking to revoke sharing privileges. The performance evaluation of our prototype demonstrates that the associated computation costs are reasonable in practice.**

***Index Terms*—medical data, attribute-based encryption, blockchain, smart contract, software defined networking**

I. Introduction

The era of cloud computing and big data analysis requires data to be shared among authorized users often from various organizations. However, in healthcare applications, electronic medical records (EMR) are usually stored in isolated hospital databases that reside in private networks due to compliance to regulations such as Health Insurance Portability and Accountability Act (HIPAA) and Health Information Technology for Economic and Clinical Health (HITECH). This data isolation introduces a significant challenge to clinical information integration and sharing.

Simply moving medical data to the cloud for storage, management, and analytics is not a panacea to the challenges. Because cloud service providers face constant internal and external security threats [1], [2], outsourcing sensitive health data to cloud without further enforcement of security and privacy protection will undoubtedly add its leakage risks.

Some cloud service providers (CSP) such as Amazon, Google, and Microsoft, proposed HIPAA compliant cloud service for medical information management. However, security and privacy issues can become increasingly complicated. Take Amazon HIPAA cloud [3] for example, they provide key management service for customers with encryption requirements for protected health information (PHI), which implies that cloud administrators have the chance to "touch" the key and to decrypt data. An option is to let hospitals encrypt data before uploading and maintain keys themselves. However, this approach leaves the challenging task of key management to hospitals, which limits its scalability on data sharing across a large scale of healthcare providers and medical institutes.

HIPAA compliance for a cloud service mainly means that the service has been audited and certified by an independent institute against HIPAA rules. It does not mean that internal technical architecture and data management mechanisms of different HIPAA cloud services will be compatible with each other. Hence, the interoperability problem that deters clinical data integration still exists, which turns various hospitals into isolated information islands [4]. For instance, when a patient gets treatment in a hospital, how can the current attending doctor get his past medical records stored in other hospitals, especially when the hospitals are on different medical record systems?

Generally, existing cyber-infrastructures for healthcare, including traditional single domain-based medical systems and emerging cloud-based solutions, mainly face the following obstacles.

Data interoperability. The shift from traditional enclosed healthcare to a more holistic and shared healthcare demands that various stakeholders work within a collaborative platform where data can be securely exchanged and shared. Existing healthcare infrastructure built in an enclosed domain is facing the difficulty of managing rapidly increasing silos of health information. Therefore, how to efficiently integrate these insular medical databases from various hospitals without violating privacy regulations becomes a difficult problem [5].

Security. Security should protect medical data in transit and at rest, so that data confidentiality, integrity, and availability can be maintained. For data in transit, currently, Transport Layer Security (TLS) protocol can be used to guarantee the communication security. For data at rest, cryptography primitives such as data encryption, digital signature, and access control mechanisms can ensure secure data access in a single domain. However, due to the HIPAA compliance requirements, how to provide privacy-preserving access control and secure sharing in a statewide or nationwide scale by integrating all hospitals still remains a challenging task.

Corresponding author: yan_luo@uml.edu

Privacy. Privacy is a closely-related concept to security but has its own concentrations, i.e., it assures that personal information are collected, used, protected and destroyed legally and fairly. For medical data, the restrictions enforced by ePHI-related regulations (e.g., HIPAA), raise new concerns for stakeholders. The regulations require all ePHI-related activities, across the entirety of data storage, transfer, and provision, to consistently abide by security and privacy rules.

1) Motivation and Contributions: To address aforementioned challenges, we need new computing paradigms to facilitate medical data sharing and collaborative usage in a privacy preserving fashion. Recently, blockchain has gained much attention for its appealing advantages such as decentralization, tamper-resistance, transparency, enhanced security, and traceability, which make it possible to shed new lights on secure sharing of medical information.

We proposed a new approach seamlessly based on synergetic components including attribute-based encryption, privacy classification, blockchain technology, and software-defined networking (SDN) to achieve secure and privacy-preserving sharing of clinical information among various care-providers. The goal of this research is to effectively connect geo-scattered hospitals and clinical research centers on a large scale to provide secure and privacy-preserving medical data sharing. Specifically, our contribution mainly lies in following aspects.

- We adopt Multi-Authority Attribute-Based Encryption (MA-ABE) scheme to enforce fine-grained access control, where rich and expressive access policy can be specified with attributes assigned to an authorized user. In addition, we devise a data privacy classification policy to differentiate multiple portions of a patient's medical records to enforce privacy protection.
- We use Ethereum blockchain to connect geo-scattered hospitals and HIPAA clouds and design smart contracts to accomplish the business logic of secure and interoperable clinical data sharing.
- To address the revocation inefficiency of MA-ABE, we deploy a software SDN layer under the blockchain to determine the final connectivity of a user to a medical database, which is based on a white list (authorized users) stored on the blockchain.

The rest of the paper is organized as follows. Section II introduces the background of this research. Section III elaborates the design of the proposed framework. We analyze its fulfillment of security and privacy requirements in Section IV. The performance evaluation results are presented in Section V. Finally, the paper is concluded in Section VI.

II. Background and Preliminaries

A. Blockchain and Smart Contract

Blockchain is a decentralized, public and immutable ledger where transactions are stored in chained blocks without the existence of a trusted central authority. As a new computing paradigm, blockchain possesses several intriguing characteristics. Firstly, the chain of blocks are immutable due to the behind consensus mechanism. An attacker has to control more than 51% computing resources to enforce a successful double-spending attack, which is hard to achieve in a completely decentralized setting. Secondly, embedded cryptographic mechanisms such as Merkle hash tree, chained hash, and digital signatures also ensure the integrity of on-chain data. Thirdly, blockchain itself serves as a shared ledger distributed to all participating nodes in the peer-to-peer network, which greatly reduces the risk of single-point-of-failures.

Bitcoin, as a famous blockchain-based cryptocurrency, applies Proof of Work (PoW) to achieve network consensus [6], while Ethereum uses a combination of Proof of Work and Proof of Stake [7]. Both strategies require participating nodes to add blocks at a certain cost, either at the expense of computation or capital.

However, the script language embedded in Bitcoin is not Turing-complete, hence it is difficult to extend Bitcoin to support various applications. It was not until 2015 when Ethereum pioneered to instantiate the "Smart Contract" concept that it becomes a reality to build various decentralized applications upon blockchain. Smart contracts are computer programs running atop blockchain that automatically execute whenever certain conditions are met, which leverage user programmed algorithms to fully customize conditions that determine when to execute or trigger other conditions. This technique broadens the scope of blockchain beyond cryptocurrency to other fields including healthcare data management.

B. Attribute-Based Encryption

In many applications, there is the need to share data according to a specific policy without prior knowledge of who will be the data receiver. Suppose a patient wants to share his medical records only with a user who has the attribute of "PHYSICIAN" issued by a medical organization and the attribute "RESEARCHER" issued by a clinical research institute. With attribute-based encryption [8], the patient can define an access policy ("PHYSICIAN" and "RESEARCHER") and encrypt his medical records with this policy, so that only users with attributes matching this access policy can decrypt the records.

Attribute-based encryption is a promising cryptographic technique for access control of encrypted data. Generally, it can be divided into two kinds, key-policy attribute-based encryption (KP-ABE) [9] where keys are associated with access policies and ciphertext is associated with an attributes set; and ciphertext-policy attribute-based encryption (CP-ABE) [9] where keys are associated with an attributes set and ciphertext is associated with access policies. In both schemes, a central authority is required to issue and validate private keys, hence they are not suitable for distributed environment where data sharing are across different administrative domains.

To address the single authority problem in existing ABE schemes, Multi-Authority Attribute-Based Encryption(MA-ABE) [10]–[12]schemes were proposed, where no central authority is needed and collusion resistance is guaranteed. In our design, we adopt decentralized MA-ABE scheme by

Lewko el. al [12] into our framework to accomplish the task of fine-grained access control and secure key storage.

C. Software Defined Networking

Health-care systems are usually built on a hospital's private network, where virtual private netwok (VPN) technology are used to provide access to medical databases. However, current MA-ABE schemes have inborn limitations in user revocation [10], [12], [13]. To address this problem, we design a workaround by introducing software defined networking (SDN) into our architecture.

By separating the control and data planes with an SDN controller, provisioning devices is automated including services, protocols, and security policies. With SDN, a centralized software program (SDN controller) can work as the "brain" to control behavior of the entire network. Hence SDN controllers can be programmed to determine the path of network packets across a network of switches, and further to provide efficient rule-based network control.

Nowadays, the rapid increase in the number and diversity of smart end devices has raised the issues of flexibility, efficiency, availability, security, and scalability within the network. Some research [14] proposed to combine blockchain technology with SDN to design a secure, scalable, and efficient network architecture, where blockchain can be used to securely verify the version of a flow rule table and validate its correctness. This also inspires us to use SDN controller to manage underlying network paths for medical data access.

D. Prior Art

Recently, some blockchain based approaches [15]–[20] are proposed to address the problem of secure medical data sharing.

Zyskind et al. [15] proposed to use blockchain to provide secure and privacy-preserving data share among mobile users and service providers. MedRec [17] firstly proposed to use Ethereum blockchain and smart contract to securely manage and share medical information. However, these schemes store plain-text medical information in private databases. An internal technical member can easily touch the data, which makes the confidentiality of information at rest difficult to guarantee. Similar to [17], MedBlock [20] proposed to store encrypted summary data and data hashes on blockchain to enforce access control and preserve data privacy.

Q. Xia el al. proposed BBDS [21], a high-level blockchain-based framework that permits data users and owners to access medical records from a shared repository. However, their secure sharing of sensitive medical information is limited to invited and verified users. The authors also proposed MedShare [18], a similar blockchain-based framework for medical data sharing that provides data provenance, auditing, and control in cloud repositories among healthcare providers.

K. Peterson el. al [16] proposed a blockchain-based approach for cross-institutional health information sharing. They designed new transaction and block structures to enable secure access of fast healthcare interoperability resources (FHIR) that

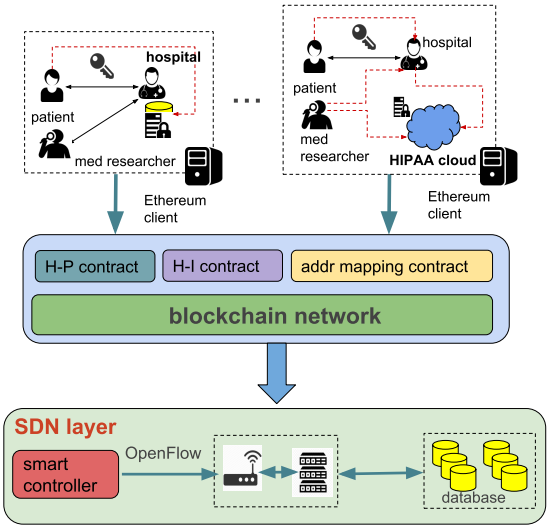


Fig. 1. Architecture

were stored off-chain. A proof-of-interoperability concept was proposed in their consensus mechanism to ensure transaction data be in conformance with FHIR structural and semantic constraints. However, the paper did not mention how the medical data are organized, stored, and accessed.

In addition, some studies propose to combine blockchain with more advanced and complicated cryptographic primitives to provide fine-grained access control and privacy protection. Liu et al. [22] proposed using ciphertext-policy attrubute-based signcryption to provide fine-grained access control and secure sharing of Personal Health Records (PHRs). Guo el al. [23] proposed to combine blockchain with a multi-authority attribute-based signature scheme to secure the storage and access of electronic health records. However, their scheme encapsulates and stores health records in on-chain blocks, which makes its scalability a troublesome problem.

Our work focuses on connecting scattered hospitals through blockchain, and using attribute-based encryption scheme and privacy control policy to provide secure data sharing. In our work, we store critical metadata with limited size on the chain, while medical information are encrypted and stored in hospitals' private databases.

III. Framework Design

A. System, Trust and Threat Model

The system involves different parties as participants, as illustrated in Figure 1.

1) **Patient** is the medical data owner and relies on care-providers to store and manage his data.
2) **Care-provider** stores and manages patient's encrypted medical information. In this sense, a care-provider can be viewed as a delegator of its patients.
3) **HIPAA cloud** provides HIPAA compliant storage service to some care-providers.
4) **Third-party institute** is a medical research institution that needs to analyze a large amount of clinical data to

conduct their research. It should gain patients' approval before obtaining their personal medical records.

Patients are owners of their medical records, however, they have to rely on hospitals to manage and store their information. In this setting, patients have almost fully trust toward hospitals. On the other hand, hospitals suffer from the trivial and troublesome task of medical data management, therefore some of them may decide to outsource their data to HIPAA clouds for storage and management. Hence, for a considerable period of time, there exist different design choices: some hospitals still use their own private data centers to store medical data while other hospitals will outsource their data to HIPAA compliant clouds.

For hospitals having moved to HIPAA clouds, it is worth noting that hospitals and HIPAA cloud base their cooperation on commercial contracts and law regulations (e.g. HIPAA and HITECH). We assume the cloud is semi-trusted even it is HIPAA compliant. This is due to following reasons:

1) A cloud service provider in essence is a third-party company authorized by hospitals to store and manage their clinical data. Even it is HIPAA compliant, the cloud can not guarantee that its technical staff will never misuse or leak medical information, let alone the external security attacks.
2) From the perspective of patients, they can place full trust on hospitals but not clouds. In this sense, we assume that HIPAA cloud behaves appropriately most of the time, but it is possible to be engaged in behaviors violating security and privacy rules due to administrative errors or attacks.

Finally, third party institutes and users may try to access more information beyond their privileges, e.g., a pharmacy cooperation may want to get prescriptions of patients for marketing reasons. Therefore, strict access control should be enforced to guarantee a user always get what he is authorized to access.

B. Framework Overview

Our aim is to effectively connect all scattered health-care providers to provide medical data sharing without violating HIPAA compliance. To guarantee data confidentiality, we encrypt medical information with symmetric encryption schemes and secure the key storage with attribute-based encryption scheme. Furthermore, by combining ABE with our privacy control policy, flexible access policy can be pre-defined by patients to allow who can access what part of their medical information.

As depicted in Figure 1, hospitals are delegators of patients to securely manage their clinical data. Any third-party medical researcher gains their authorization from patients and then access data through hospitals or HIPAA clouds. HIPAA clouds may adopt their own security mechanisms, which are independent to ours. In our framework, we view HIPAA clouds as black boxes for storage.

Considering the fact that hospitals and HIPAA clouds are not immune to attacks and intrusions, encrypting medical

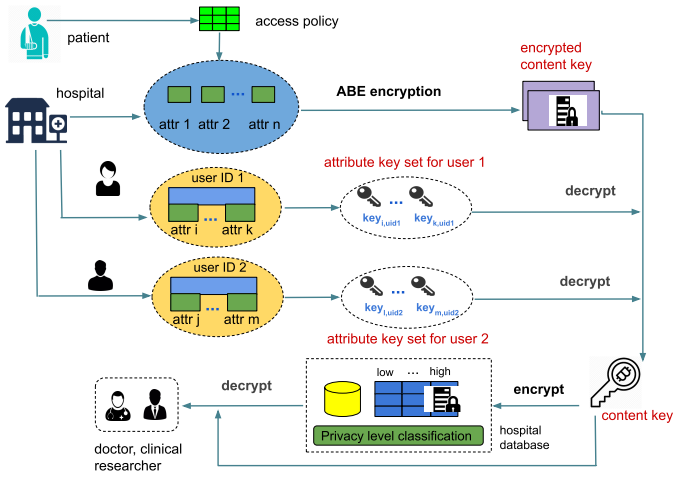


Fig. 2. Decentralized attribute-based encryption of content keys

information is a good choice to guarantee data security at rest. However, simple encryption to millions of patients' clinical data will leave key management a difficult task, which also provides limited granularity of access control. To guarantee confidentiality and fine-grained access control simultaneously, we decide to introduce the multi-authority attribute based encryption scheme [12] into our design to secure the storage of the content key, which is used to encrypt(decrypt) a patient's medical data. Thus, we can provide rich and fine control to users' access privileges. Furthermore, encrypting a patient's medical records with a single content key makes it easy to classify medical information into multiple parts with different privacy levels, as will be elaborated in the following section.

According to Figure 1, we adopt Ethereum blockchain to integrate all hospitals and HIPAA clouds to share clinical information. Each hospital or HIPAA cloud can be viewed as an independent administrative domain, data sharing across multiple domains would rely on the executions and interactions of corresponding smart contracts. By designing smart contracts to implement the business logic of clinical information management across various provider domains, data access behavior can be recorded in a tamper-evident and traceable way.

Finally, SDN is deployed in each health-care provider's domain, which controls the final connectivity for all data access behavior. Each hospital will be equipped with a SDN smart controller that is a software component residing in a server, who is responsible for constructing an effective network path for data access according to policies defined by smart contracts. Hence, the SDN layer provides a second-level protection of medical data to ensure that only authorized users can connect to databases.

C. Access Control

To guarantee data confidentiality and privacy, one feasible way is to encrypt data and keep keys in a safe place. However,

TABLE I
PRIVACY LEVEL DEFINITIONS

Institute Category	Privacy Level Description
Pharmacy Company	PL: low
Insurance Company	PL: medium
Medical School	PL: high
Hospital A	PL: complete
Hospital B	PL: complete

TABLE II
NOTATIONS AND KEYS

Notation	Description
A_{def}	default access policy
K_E	encryption key for medical information
$E(M, K_E)$	encrypt data M with content key K_E
$E_{ABE}(m, A)$	attribute-based encryption of message m
$addr$	Ethereum address
$K_{i,addr}$	secret attribute key of attribute i

scalability of such measures is limited. In a blockchain network with millions of nodes involved, key management and distribution will be a tedious and difficult task to accomplish. Thus we need more advanced cryptographic primitive to address this problem.

We adopt a two-level encryption structure for data protection. Firstly, a decentralized multi-authority CP-ABE scheme [12] with rich access policy customization ability is adopted to encrypt and securely manage content keys. Secondly, content keys are used to encrypt patients' medical information. Figure 2 depicts such a two-level encryption process in our design.

In our setting, each hospital or HIPAA cloud is an independent administrative domain, which corresponds to an authority in MA-ABE scheme. Each authority is responsible for issuing and validating attribute secret keys in his domain. According to the definitions of ABE, encryption takes in an access policy as one of its input parameters, which is defined and assigned by patients. Then a hospital enforces attribute-based encryption to produce the ciphertext, namely, the encryption results of a content key. On the other hand, by defining his own access policy, every patient can enforce and implement customized access control according to his privacy requirements. If a patient is unwilling to define his own access policy, a default policy A_{def} will be provided by the system, which is defined according to past experience and HIPAA regulations.

1) Privacy Level Classification: Generally, a patient's medical information contain multiple parts with different privacy requirements. For example, information which can be used to distinguish or trace an individual's identity, such as name and social security number, may be highly sensitive so that the patient is unlikely to reveal. While for general information with low sensitivity such as diagnosis records and treatment methods, the patient might allow it to be shared with others. Hence, a multi-level privacy classification mechanism is needed to enforce privacy control on medical information.

We format a patient's medical information as records containing multiple data fields. Let $R_i = (d_1, d_2, ..., d_n)$ denote a medical record. According to different sensitivities of data fields, a patient can assign different privacy levels (low, medium, high, complete) to data fields contained in his medical records. Obviously, higher privacy level requires higher privilege. In storage, every data field will be attached with a tag field whose value is a small positive integer indicating the privacy level of the data field.

Such a structure can provide multi-level privacy control for medical information without incurring much additional overhead. Assume a byte is allocated for each tag field, then the additional storage overhead of tag values for each patient is several bytes, which is negligible. Let's assume the medical records are divided into multiple parts with different privacy levels $\{L_1, L_2, ..., L_k\}$, and a user's privacy level being L_u. Then only data fields labelled with a tag $L_i (1 \leq i \leq k)$ satisfying $L_i \leq L_u$ can be accessed by the user. For each third party institute, whose privacy level value is defined by patients in hospital-patient smart contracts stored on the blockchain. Such a mechanism can let an authorized user with a specific privacy level only access those record fields he is allowed to. Talbe I depicts an example of a patient's privacy control policy, where hospitals from which the patient receives treatment (e.g., hopital A and B) can access all his medical records. Other institutes such as insurance or pharmacy companies can only access a part of information the patient allows them to read.

2) MA-ABE Encryption: Each authority in our architecture has its own set of secret-public key pair (SK_i, PK_i) for each attribute i in its domain. For each authorized user with a specified attribute set $a_1, a_2, ..., a_k$, the authority needs to generate a corresponding attribute private key $K_{a_i,GID}$ for each attribute a_i in the set, as illustrated in Figure 2. It is worth noting that GID is a globally unique identifier, it is used as a "linchpin" for tying a user's attribute private keys together, so that collusion problem can be avoided in multi-authority setting. This idea is first proposed by Chase [10] and then used in Lewko's MA-ABE scheme [12]. One possible candidate for GID is a user's social security number, as suggested by Chase [10]. In our framework, we use a user's Ethereum address as a candidate for GID, thus we can leave the task of id acquiring and validation to Ethereum. Table II depicts notations and their descriptions in our framwork.

Each authority, i.e., a hospital, a care provider, or a HIPAA cloud service provider in our architecture, should run an $AuthoritySetup()$ algorithm to create a public-private key pair for each attribute it manages according to the following algorithms.

$AuthoritySetup(GP) \rightarrow PK, SK$ For each attribute i, the authority (hospital or care-provider) chooses two random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and computes $PK = \{e(g_1, g_1)^{\alpha_i}, g_1^{y_i} \forall i\}$. It keeps $SK = \{\alpha_i, y_i \forall i\}$ as its secret keys.

Then, an authority should run algorithm $KeyGen()$ to

create attribute private keys for authorized users.

$KeyGen(addr, GP, i, SK) \rightarrow K_{i,addr}$. To create a key for $addr$ for attribute i, an authority computes:

$$K_{i,addr} = g_1^{\alpha_i} H(addr)^{y_i} \quad (1)$$

where g_1 is a public parameter which refers to a group generator, and $addr$ refers to the Ethereum address of a user, we use it to replace the global id GID in original MA-ABE scheme.

On decryption, only when the set of attribute private keys that a user holds satisfy the access policy contained in the ciphertext can the user successfully decrypt the ciphertext to recover the original message. We omit the details of encryption and decryption algorithms here, we recommend readers to find more details in Lewko's MA-ABE scheme [12].

D. Smart Contract Design

Ethereum was designed as a fee system where any amount of consumption on computation, bandwidth and storage needs to pay proportional gas fee (Ethereum currency), which is to prevent hostile infinite loops in DoS (denial of service) attacks. Storing data on blockchain is expensive. According to the Ethereum yellow paper [24], storing a kilobyte cost 640 thousand gas, which amounts to $3.4 even at a relatively low gas price of 20Gwei (Ethereum currency, 1 Ether = 10^9 Gwei) and with Ether at $272 recently. Hence, it is impractical and expensive to store detailed clinical information of millions of patients on blockchain, instead, only a very tiny subset of critical metadata can be stored on chain.

We assume the existence of a global certificate authority (CA), who is responsible for issuing credentials to hospitals and third-party medical institutes. With these credentials, hospitals and medical institutes can be authenticated and participate in the blockchain network at the initial setup phase. Afterwards, each hospital is regarded as an authority in an independent domain and responsible for managing medical information in its local database or through a third-party HIPAA compliant cloud. With blockchain and smart contracts, geo-scattered hospitals and private HIPAA clouds are connected, medical information are shared in a secure and controlled way.

To implement the business logic of clinical data management, we designed four smart contracts.

(1) Address Mapping Smart Contract (AMSC), registers the participants in the blockchain network and holds a network whitelist table. AMSC contract establishes three mappings (hash tables) to store Ethereum addresses of the involved participants: 1) A global CA maintains the hospital mapping where each hospital or will be authenticated and registered before participating in the blockchain. 2) Each hospital register their patients and authorized institutes in the remaining two mappings in AMSC.

The whitelist mapping maintained by the hospitals, contains network path information for an authorized institute to access a hospital's private database. The SDN layer leverages this whitelist to control the final connectivity to data access.

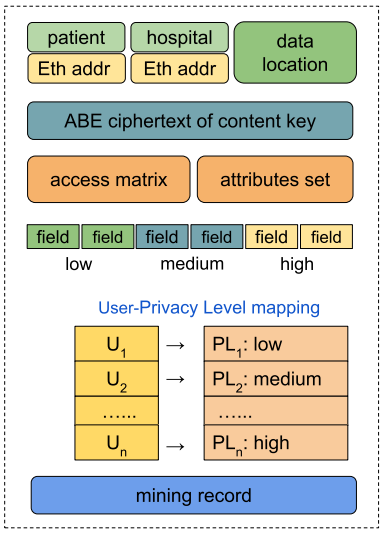


Fig. 3. Hospital patient smart contract

The AMSC contract performs authentication by checking the Ethereum address to ensure each hospital can only change data content belongs to it. Thus, we can just store a global whitelist in the contract instead of storing one list for each hospital.

(2) Hospital-Patient Smart Contract (HPSC), defines the relationship between a patient and a hospital, where the hospital stores and maintains patients' clinical information. As illustrated in Figure 3, we store the ABE cipher-text of the encrypted content key and the access policy in the contract. This contract is generated by the cooperation of both parties when the patient visits the hospital to get treatment for the first time. On generating the contract, the patient has to provide an access policy that defines his personal privacy requirements, or use a default one provided by the hospital. Meanwhile, the patient gives a privacy level tag for each third-party institute. A simple way to do this is to classify the institutes by different categories (e.g., insurance companies has a medium level).

(3) Hospital-Institute Smart Contract (HISC), describes relationships between a hospital and a patient or a medical institute. A patient's medical records are encrypted with a content key, which is encrypted using attribute-based encryption. Then a user has to decrypt the ABE cipher-text $E_{ABE}(m, A)$ to recover the content key K_E. During this process, the hospital has to compute and send required attribute private keys $K_{i,addr} = g_1^{\alpha_i} H(addr)^{y_i}$ to the user. With these attribute keys, the user is able to recover the content key if his possessed attribute key sets satisfy the access policy pre-defined by the data owner.

(4) Access Request Smart Contract (ARSC), allows institutes to submit requests for patients' medical records. ARSC is also used to authenticate a third party institute when it firstly contacts a hospital to request for medical information. As we have mentioned, all hospitals and institutes have a credential or certificate issued by a global CA. Hence during the execution of ACSC, the hospital needs to verify the credential of the institute to authenticate its identity. If succeed, the hospital

will create a HISC contract to define their relationship.

Discussion. The existence of a CA is necessary for the setup phase when all hospitals and institutes participate in the blockchain, since each entity should be authenticated during this initial step. This assumption is much weaker than the assumption of the existence of a central authority in single domain based ABE schemes [9] or MA-ABE [10], where the central authority need to integrate secret keys from different attribute authorities. This is also the reason of our adoption of Lewko's MA-ABE scheme [12], where such an central authority is not necessary.

E. SDN layer

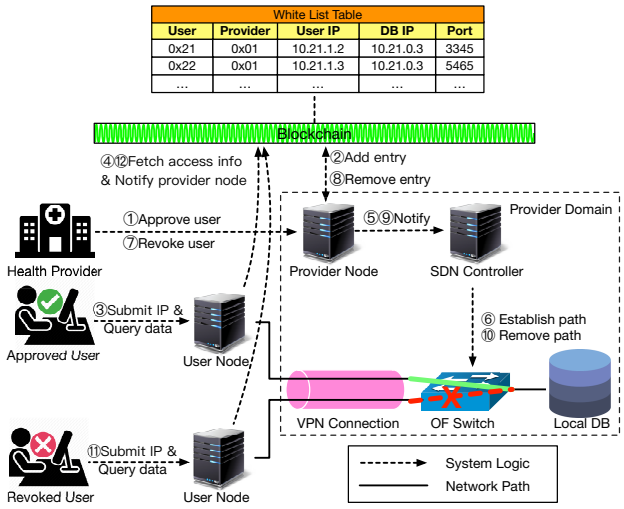


Fig. 4. SDN workflow

The whitelist stored in the AMSC contract is used by the SDN layer to determine a user's final connectivity to a hospital's database. Since the adopted MA-ABE scheme does not support user revocation, we rely the SDN controller to deny a revoked user's connectivity to a medical database according to the whitelist.

Figure 4 depicts the workflow of the SDN layer. When a care provider approves a user's access request, it needs to generate an access string that consists of an url indicating data location and a SQL query string, which will be stored in HISC. Meanwhile, the provider will create a virtual private network (VPN) credential associated with a static VPN IP for the user, so that he can connect to the provider's network from external network. Then, the provider will add an entry in the whitelist where the user's static IP and Ethereum address will be binded and stored. Step 1 and 2 describe this process.

Step 3 to step 6 describe the process of how the SDN controller provides connectivity for an approved user. When an approved user connects to the provider's network using VPN and queries the data, the user node firstly fetches the aforementioned access string from corresponding HISC and notifies the provider node. Then the provider node requires the SDN controller to apply OpenFlow [25] rule to the underlying network device (i.e., the openflow switche in Figure 4), which is to provide final connectivity for the approved user. Afterwards, the approved user can send access string to the destination database to get required data.

Step 7 to step 12 describe the process of user revocation. When a provider revokes a user, the provider node simply deletes the user's entry from the whitelist and notifies it to the SDN controller. Then the controller removes this path from the network. Afterwards, there is no matching rule in the whitelist for the revoked user. Hence even the user still has the access string and can connect to the provider's network, he cannot connect to the destination database.

Whitelist. The whitelist is used by a SDN controller to determine whether to provide network connectivity for a specific user. As illustrated in Figure 4, each entry in the whitelist contains Ethereum addresses of a care-provider and an approved user, a static VPN IP for the user to connect to the care provider's network, and the destination IP and port number of the provider's database. An approved user has an entry in the whitelist while a revoked user's entry will be deleted from the list.

The whitelist is stored on-chain in the AMSC contract. The update function embedded in AMSC guarantees that a provider can only update entries containing the provider's Ethereum address, namely, users belong to the provider's domain. That means a provider cannot modify another provider's content in the whitelist. Another option is to store a local whitelist in each provider's domain, however, it may make the node storing the whitelist a single point of failure. By storing a global whitelist on chain, we can guarantee its integrity via the tamper-evident characteristic of blockchain. In implementation, assume a 1000-user scale and each user has a 20-byte Ethereum address, the storage overhead is about 40KB, which is acceptable since it is the only global metadata for SDN-based access authentication and authorization.

The adopted Multi-Authority Attribute-Based Encryption scheme does not provide user revocation functionality. To revoke a user, we have to generate a new content key to re-encrypt medical information and re-compute the ABE ciphertext of the new key, which brings heavy computation cost. Hence we devise our own revocation mechanism based on SDN controller. Such a straightforward policy is easy to implement and avoids re-encryption. With SDN controller managing network accesses, even a hacker can get the access string, his network access to data will be blocked by the OpenFlow network device. In this sense, the SDN technology deployed within each hospital provides a second-level protection.

Finally, with the increasing amount of medical data, a hospital will create more databases to store data. These new databases can simply be connected to the SDN network and let SDN controller take charge of the network access to them. Otherwise, new access configuration have to be added to the network device manually. Hence, the SDN layer provides a scalable network architecture.

IV. Security Analysis

In this section, we briefly analyze the fulfillment of our design goals on security and privacy.

Confidentiality. We adopt a two-level encryption policy to guarantee the confidentiality of medical data. To recover encrypted medical information, an attacker has to decrypt the ABE cipher to get the content key. One way is to get enough attribute private keys reflected by an legitimate access policy from an authorized user, which contradicts to the security assumption in ABE. The other way is to break the ABE scheme, whose security has been proved under the random oracle model. Hence, the confidentiality guarantee is dependent on the security of the adopted MA-ABE scheme.

Integrity. Each medical record is attached with a signature signed with the patient's private key. Any other party can verify its integrity with the public key. Due to the security of digital signatures, an unauthorized user cannot forge a valid signature of a medical record without the signing key.

The whitelist is stored on chain and maintained by all involved care-providers. The integrity of the whitelist is guaranteed by the blockchain and the consensus protocol behind, since it is stored in the AMSC contract and further included in a mined block.

Privacy. Our privacy classification policy for record fields enables fine-grained sensitivity differentiation. Combined with user-customized privacy policy, it allows a patient to decide which part of his medical information can be accessed by what kind of users. In implementation, data required by an authorized user will be filtered according to the user's privacy level before being sent to him. As we have mentioned in Section III-A, we assume hospitals are trustworthy, hence a patient's privacy control depends on the proper execution of his privacy policy by the hospital. However, in a real world setting, it is possible for two parties collude to cheat another party (e.g., a hospital may send more data than the user is allowed to access, thus violate the privacy policy defined by the patient). How to detect or prevent such collusions is beyond the scope of this paper, we will address it in our future work.

V. Performance Evaluation

A. Software Components

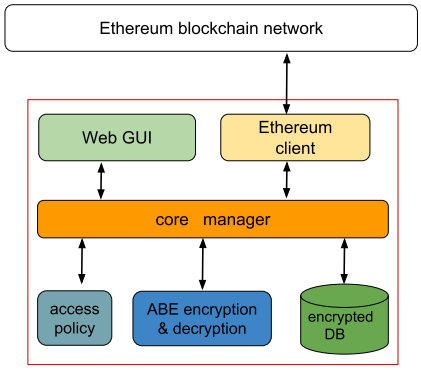


Fig. 5. Node Components

We have implemented a prototype for our framework. Our system architecture mainly consists of five components: Web GUI, Ethereum Client, Core Manager, ABE Encryptor and Decryptor, access policy manager, as illustrated in Figure 5.

Web GUI is a browser based interface provided to all users, by which a hospital can set up electronic healthcare file for a patient and enforce attribute-based encryption, a patient can define his access and privacy control policy, and a third-party user can request to access patients' medical information according to his permission. An Ethereum client is responsible for connecting to the peer-to-peer blockchain network, generating and encoding new transactions, and executing smart contracts. Every provider node will maintain a encrypted database where patients' medical records are stored. A core manager component is responsible for coordinating and transfer messages among other components, e.g., translation of GUI operations into arguments to ABE encryption and decryption, generation of access strings to encrypted database, and interaction with Ethereum clients.

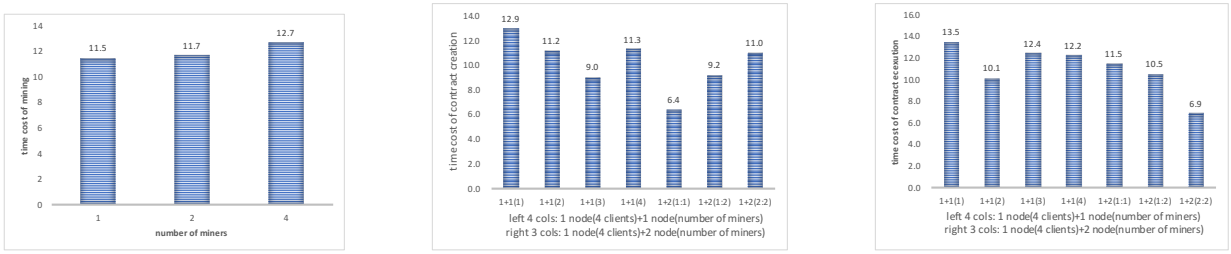
B. Smart Contract Execution Time

We use Ethereum Solidity to implement our smart contracts(AMSC, HPSC, HISC, and ARSC). Among them, AMSC is to keep an Ethereum address mapping between hospitals, patients and institutes. It also stores the whitelist used by the SDN layer in our architecture. HPSC indicates the relationship creation between a hospital and a patient. To create this contract, the patient needs to define his own ABE attribute set and classify privacy levels of his information, or adopts a default policy.

We divide the smart contract test into two steps. Firstly, we run an Ethereum client node in a server to mine blocks without including any transactions. Its mining time can be viewed as a benchmark. As illustrated in Figure6(a), we can see that the number of miners deployed in a node has no obvious effect on the mining time, three columns in the figure are very close to each other, with their values fluctuating around 12 seconds.

Secondly, to further test our business logic with smart contract creation and execution, we deployed five servers in our local area network. Among them, one server is deployed with four Ethereum Geth clients to denote a provider, a patient, a institute, and a CA; the other two servers are equipped with mining clients with their number varying from 1 to 4. It is worth noting that the contract creation or execution means the time period from creating or executing a smart contract to the point when it is included in a successfully mined block. We found it is mainly decided by the block mining operation, which is further decided by the PoW consensus in current Ethereum platform. As Figure 6(b) depicts, the left four columns show the time of contract creation when one server is used to mine blocks (number of miners varies from 1 to 4); the right three columns denotes the time when two servers are used to mine (number of miners on two servers vary from 1:1 to 2:2).

The execution of smart contracts takes between 10 and 20 seconds, which accord with the average mining overhead of an



(a) Benchmark of block mining time (b) Smart contract creation speed (c) Smart contract execution speed

Fig. 6. Time cost of smart contract creation and execution (block mining)

Ethereum block. The two exceptions are creation time of the "1+2(1:1)" column in Figure 6(b) and the execution time of the "1+2(2:2)" column in Figure 6(c). This is possible, because our test environment is a lab LAN, where the propagation time for a transaction to be broadcast to the majority of peers can almost be omitted. The average block mining time of our test is 10.6 seconds, which is 3.7 seconds less than current Ethereum block mining time.

Finally, our test result includes 220 trials of smart contract execution and block mining. Among these tests, about 13.2% spends more than 20 seconds and 86.8% spends less than 20 seconds. This is reasonable. Because the 10-20 seconds mining range refers to the average mining time, there are blocks whose mining time exceeded this range(e.g., mining time between late September to mid October in 2017 is near 30 seconds [26]).

C. ABE overhead

TABLE III
TEST ENVIRONMENT

Item	Specifications
Host CPU	2 x Intel E5-2643 v3 @3.40 GHz
CPU cores	6 core
Host Memory	32 GB ECC Memory
Disk	1024 GB at 7200 RPM
Host OS	Ubuntu 16.04.4
Charm crypto lib	0.43

To evaluate the overhead brought by MA-ABE, we use Python 3 to implement a prototype based on the Charm-crypto library (version 0.43). We choose the "SS512" option in Charm to initialize a group in the elliptic curve setting, which represents a symmetric curve with a 512-bit base field. The test environment is depicted in Table III.

We have tested the overhead of $AuthoritySetup$, $Encrypt$ and $Decrypt$. Figure 7(a) shows the setup overhead for a single authority (a care provider), which is almost linear to the number of attributes managed by the authority. This is because the main task of setup is to generate a public-private key pair for each attribute (e.g., it takes 75 milliseconds to generate key pairs for 30 attributes).

While for ABE encryption and decryption, considering the fact that their overhead are mainly decided by the complexity of the involved access policy, we test the overhead by providing different access policies. The boolean formulas defined in a policy is in the form of $(C_1\ or\ C_2)\ and\ (C_3\ or\ C_4)\ ...\ (C_{2i+1}\ or\ C_{2(i+1)})$, each $(C_{2i+1}\ or\ C_{2(i+1)})$ is regarded as a subformula. Figure 7(b) and 7(c) denote the overhead of ABE encryption and decryption when the number of subformulas varies from 2 to 8. We can see that their overhead are almost linear to the number of subformulas contained in the access policy. But decryption spends less overhead than encryption, e.g., when the number of subformulas is set to 6, decryption needs 23 milliseconds while encryption needs 113 milliseconds.

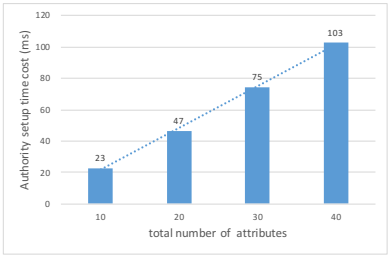
D. Discussion

The current prototype implementation is based on Ethereum due to its maturity on smart contract design and execution. However, it does not mean Ethereum is the only choice. On the contrary, the gas consumption in Ethereum for contract execution is expensive, which hinders its application in medical data sharing for its potential enormous expenditure on clinical data management. Our experiment is to test the feasibility of managing medical data sharing and integration through blockchain, which is proved acceptable in the evaluation. In the future, we may turn to permissioned blockchain such as hyperledger [27], which provides higher throughput than permissonless blockchains (e.g., Bitcoin and Ethereum) to address daily medical events. Moreover, its multi-authority nature makes it more suitable for national-scale medical data sharing by integrating scattered hospitals.

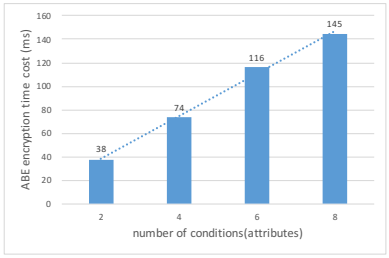
VI. CONCLUSION

This paper is an attempt to provide secure medical data sharing among various care-providers, patients and medical researchers without privacy violation. We adopt multi-authority attribute-based encryption scheme to secure the data encryption key storage with fine-grained control and rich policy customization. By combining it with a privacy level classifier, we can also enforce strict privacy control for medical information.

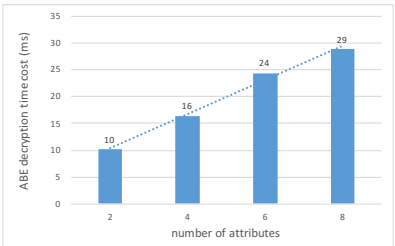
To connect scattered hospitals and care providers residing private networks and enable clinical data integration and sharing, we design smart contracts under Ethereum platform to integrate underlying components. Moreover, to provide



(a) Time cost of authority setup



(b) Time cost of ABE encryption



(c) Time cost of ABE decryption

Fig. 7. Cryptographic cost of ABE

efficient user revocation and scalable network management, we design a SDN layer in our architecture to provide network paths for authorized users when they request data from medical databases and to efficiently revoke access if needed. Performance evaluation demonstrates that our design is feasible: computation overhead brought by ABE encryption is acceptable; execution overhead of our four smart contracts and block mining time are within a reasonable range.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their reviews and insightful suggestions to improve this paper. This work is partially supported by the National Science Foundation of USA (Award No. 1547428, No. 1738965 and No. 1450996).

REFERENCES

[1] R. Walters, "Cyber attacks on us companies in 2014," *The Heritage Foundation*, vol. 4289, pp. 1–5, 2014.

[2] G. Jordi, "Linkedin data leakage: Change your password now," http://www.linkedin.com/pulse/linkedin-data-leakage-change-your-password-now-jordi-gili/, 2016.

[3] "Architecting for hipaa security and compliance on amazon web services," https://d1.awsstatic.com/whitepapers/compliance/AWS-HIPAA-Compliance-Whitepaper.pdf, 2018.

[4] Y. Luo, H. Jin, and P. Li, "A blockchain future for secure clinical data sharing: A position paper," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2019, pp. 23–27.

[5] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, pp. 61 656–61 669, 2019.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2009.

[7] Ethereum, "Proof of stake faq," https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, 2014.

[8] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2005, pp. 457–473.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[10] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography Conference*. Springer, 2007, pp. 515–534.

[11] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 121–130.

[12] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2011, pp. 568–588.

[13] K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 536–545.

[14] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, 2017.

[15] 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.

[16] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proceedings of the NIST Workshop Blockchain Healthcare*, vol. 1, 2016, pp. 1–10.

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

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

[19] H. Yang and B. Yang, "A blockchain-based approach to the secure sharing of healthcare data," in *Proceedings of the Norwegian Information Security Conference*, 2017.

[20] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, p. 136, 2018.

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

[22] L. Jianghua, H. Xinyi, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Generation Computer Systems*, vol. 52, pp. 67 – 76, 2015, special Section: Cloud Computing: Security, Privacy and Practice.

[23] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 776, no. 99, pp. 1–12, 2018.

[24] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger. ethereum project yellow paper 151 (2014)," 2014.

[25] B. Pfaff, B. Heller, D. Talayco, D. Erickson, G. Gibb, G. Appenzeller, J. Tourrilhes, J. Pettit, K. Yap, M. Casado *et al.*, "Openflow switch specification," 2009.

[26] https://etherscan.io/chart/blocktime, 2018.

[27] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 30.