An Attribute-Based Distributed Data Sharing Scheme

Ehab Zaghloul, Tongtong Li and Jian Ren

***Abstract*—Patients rely on their public health records shared among medical institutions to receive the appropriate treatment they require. They must completely trust that these institutions will secure their records, protect their privacy, and efficiently share them when requested by other institutions. Unfortunately, medical institutions cannot fully be trusted for several reasons. First, patient records are stored on the servers of the medical institutions which could result in security issues and also a single point of failure. Second, centralized storage may also result in privacy concerns if records are incorrectly shared or leaked. Third, institutions may purposely delay sharing patient records for competitive reasons. To address these issues, we propose an attribute-based distributed data sharing scheme for patients to control how their records are shared. The distributed file sharing can effectively prevent the single point of failure and ensure data availability upon its request. Moreover, patients are also given the capability of selectively sharing their records for privacy protection. Our analysis shows that while ensuring attribute-based sharing of medical records, the proposed scheme can also work with the peer-to-peer distributed network storage such as InterPlanetary File System (IPFS) to improve efficient data retrieval.**

***Index Terms*—Public health record, single point of failure, attribute-based, distributed data.**

I. Introduction

Lack of the appropriate infrastructure to timely access the critical patient medical information from the medical institutions may lead to serious complications of patient treatment or even deaths. Today, patient record information can exist in various formats, including traditional paper systems or various digital record systems, based on the capabilities of the medical institutions. Medical institutions are encouraged to share the records of their patients via electronic Health Information Exchanges (HIE) [1]. The United States government has led the effort to computerize healthcare data by providing billions of dollars in funding with the intention that moving to computerized systems will improve patient care and reduce costs. However, most of the available exchange platforms are centralized, putting the records of these patients at risk. Additionally, they require the competing medical institutions to collaborate in order to share medical records when needed. This presents not only serious security concerns but is also a major roadblock in the standardization of systems as health IT developers can knowingly hinder information or block record transfer in a way that inhibits healthcare improvement.

Besides competing systems, the healthcare system in the United States operates in a paternalistic manner, by severely limiting access of patients to their own records. This is stark contrast to the health systems of countries like Australia, where patients have increased ownership of their own healthcare records [2]. Allowing health records to be controlled by the patients can resolve issues of record blocking or incompatibility among systems in health record sharing.

A secure, private and efficient data sharing scheme should allow patients to share their records with certain staff members of medical institutions based on their attributes. Patients should be able to grant certain parties access to parts of their records based on predefined data sensitivity preferences. This means, each patient may have a different attitude in terms of sensitivity of their record attributes. As a result, some patients are more willing to share sensitive parts of their record more than other conservative patients. In addition to this, patients should not be concerned with the availability of their records upon being requested by the permissioned staff members of medical institutions. Records generated by any medical institution should be stored in an efficiently accessible storage space that is controlled by the patient, not the medical institutions.

In this paper, we propose a data sharing scheme that addresses these concerns. Our proposed scheme utilizes an Attribute-Based Encryption (ABE) architecture to provide the patients with fine granularity when granting permission rights to the different parts of their records. Staff members at any medical institution must possess the correct set of attributes predefined by the patient in order to obtain access to the requested parts of their record(s). In order to enhance data availability when requested, we design our scheme to store the records of the patients over a distributed storage system known as InterPlanetary File System (IPFS) [3]. By storing the records over IPFS, records are securely maintained and accessed efficiently when requested by the permissioned staff members.

The rest of this paper is organized as follows. In section II, we discuss the preliminaries to our work. Next, in section III, we present our proposed scheme. Following that, in section IV, we discuss the security analysis of our work. In section V, we present our performance evaluation of our proposed scheme. Finally, in section VI, we conclude our work.

II. Preliminaries

A. Bilinear Maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of the same prime order p. The generator of $\mathbb{G}_0$ is denoted as g. A bilinear map from $\mathbb{G}_0 \times \mathbb{G}_0$ to $\mathbb{G}_1$ is a function $e\colon \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ that satisfies the following properties:

Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$ for any $a, b \in \mathbb{Z}_p$.

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

Symmetry: $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ for any $a, b \in \mathbb{Z}_p$.

Non-degeneracy: $e(g, g) \neq 1$.

Computability: $e(g, g)$ is an efficiently computable algorithm.

B. Access Structure

An access structure [4] represents access policies for a set of individuals interested in gaining individual access to a secret. The access structure defines sets of attributes that can be possessed by a single individual to allow access to the secret. It is defined as follows:

Let $\{P_1, \ldots, P_n\}$ be a set of parties. A set of parties that can reconstruct the secret is defined as a collection. The collection is monotone meaning that, if $\mathscr{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ then $\forall B \in \mathscr{A}$ and $B \subseteq C$ implies $C \in \mathscr{A}$. An access structure is a monotone collection $\mathscr{A}$ of non-empty subsets $\{P_1, \ldots, P_n\}$, i.e., $\mathscr{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \emptyset$. The sets in $\mathscr{A}$ are called the authorized sets and the sets not in $\mathscr{A}$ are called the unauthorized sets.

In this paper, we use parties to represent the attributes. This means that an access structure $\mathscr{A}$ may consist of both authorized and unauthorized sets of attributes.

C. Leveled Access Tree $\mathcal{T}_i$

An access tree $\mathcal{T}_i$ represents an access structure that determines whether a data user can decrypt the ciphertext CT_i. A $\mathcal{T}_i$ may consist of multiple nodes. We use $x_l^i \in \mathcal{T}_i$ to represent the l^{th} node of $\mathcal{T}_i$. The non-leaf nodes of $\mathcal{T}_i$ are in the form of threshold gates, while the leaf nodes represent possible attribute values possessed by data users.

For every node $x_l^i \in \mathcal{T}_i$, a threshold value $k_{x_l^i}$ is assigned. A node in the form of an AND gate is associated with a threshold value $k_{x_l^i} = num_{x_l^i}$, where $num_{x_l^i}$ represents the number of children of node x_l^i. A node in the form of an OR gate or any leaf node representing an attribute is associated with a threshold value $k_{x_l^i} = 1$.

The root node x_1^i of each $\mathcal{T}_i$ carries a secret s. The data user that possesses the correct set of attributes can satisfy $\mathcal{T}_i$ and obtain the secret s.

D. InterPlanetary File System

InterPlanetary File System (IPFS) [3] is a distributed file system that runs over a peer-to-peer network to store and share data over the network nodes. The system can be viewed as a stack of protocols.

Identities: Nodes that wish to connect to the network must generate node identities before being able to establish connections with the connected nodes. In IPFS, node identity generation is inspired by S/Kadmelia Distributed Hash Tables (DHT) [5] where each node generates a Public Key Infrastructure (PKI) key pair. The public key of each node is processed using a cryptographic hash to create its identity which is referred to as NodeId. Before a connected node accepts an incoming connection request from another node, it checks whether the hash computation of the incoming nodes public key matches its NodeId. If both values are equal, a connection is established, otherwise, the connection request is terminated. Node generation identity also includes a Proof-of-Work (PoW) crypto-puzzle making identity generation hard and expensive, hence, this design can counter Sybil attacks.

Network: The network layer is responsible for managing the established connections between the connected peers. It does not require nodes to possess IP addresses and is designed to run on top of any network. Customized addresses may be utilized and are stored in multiaddr formatted byte strings. These strings express the details of the utilized addresses and the protocols they run over for the underlying network to interpret. For example, an SCTP/IPv4 connection could be stored in multiaddr as /ip4/10.20.30.40/sctp/1234/. The underlying network would interpret this information and manage the nodes accordingly.

Routing: The main purpose of the routing system is to locate the addresses of peer nodes and the data these nodes locally store. This data is required for nodes to share data as it is requested and is achieved by utilizing a Distributed Hash Table (DHT) inspired by S/Kademlia [5] and Coral [6]. For efficiency purposes, IPFS stores data of size 1KB or less directly on the DHT. For data with larger sizes, IPFS stores the NodeIds of the peers that store this data and can serve it when requested in the DHT.

Exchange: IPFS introduces an exchange protocol inspired by BitTorrent [7], known as BitSwap, for peers to exchange data they store with each other. To exchange the data stored by the nodes locally, each node shares a want_list (a list of data it is searching for) and a have_list (a list of data it stores) with its peers. In the event that the peers of a certain node do not store the data that the node requests, the peers begin requesting this data from their own peers in a recursive manner until it is found. To keep the nodes incentivized to share the data they store, even when they themselves do not request any data, BitSwap introduces BitSwap Credit system. This system acts as a credit-like system that keeps track of how data was previously shared between peers. Based on the credit balance between two peers, the sending peer decides whether it wants to send that requested data and expects the data to be repaid, or it ignores the request if a bad credit balance has not been resolved yet.

Objects: Data stored over IPFS is represented in an IPFS object data structure format. Depending on its size, data is divided into smaller partitions and each partition is then cryptographically hashed. The resulting digests are used to reference the locations of the corresponding partitions, hence, a content-addressable system is developed. Hashing the partitions also provides data tampering resistance and single storage of duplicated data partitions. To allow data users to recover related partitions stored over IPFS, each partition is linked to one or more other partitions forming a link between all of them. The reference links of a single partition are stored in an IPFS link array as cryptographic hashes of the target data partitions. That means, if a node gains access to a particular data partition, it can also recursively gain access to all of its children data partitions,

however, it cannot obtain access to the parent data partitions. IPFS utilizes a Merkle Directed Acyclic Graph (DAG) [8] to maintain these reference links.

Files: Inspired by GIT [9], IPFS can also model a versioned file system on top of the Merkle DAG [8]. An object in this model consists of a variable sized data block (sometimes referred to as a blob), a list containing a collection of blocks, a tree that combines the blocks and lists, and a commit which reflects a snapshot in the version history tree.

Naming: The naming system used by IPFS is referred to as InterPlanetary Naming System (IPNS). IPNS allows mutable naming to immutable data content maintained by the Merkle DAG. This means, data owners can create new versions of their data and redirect to them the reference links that pointed to the older versions. As a result, the data users will continue to use the original reference links which point to the new data version.

III. The Proposed Scheme

Assume patient p visits a medical institution m to receive medical treatment. The majority of institutions employ staff members such as administrators, technicians, nurses, physicians, pharmacists, etc., that perform one or more roles which are necessary for the patient to receive the correct treatment. After the patient interacts with various of these members and receives the appropriate treatment, one of the staff members, usually a physician, generates a medical record $\mathcal{R}$ for the patient. The record may consist of multiple attributes such as personal information, medical treatment, medications, etc. We denote the set of record attributes generated by a medical institution as $\{R_j \in \mathcal{R} \mid 1 \leq j \leq n\}$, where n is the maximum number of attributes within the record. We also denote the corresponding attribute values of the record generated for the patient as $\{a(R_j) \mid 1 \leq j \leq n\}$.

Later on, the patient visits the same/different medical institution to receive more medical treatment. The patient wishes to share his/her record(s) from the previous visit(s) with the staff members of the current medical institution to obtain a more accurate treatment. However, for privacy concerns, the patient may wish to share the record differently among the staff members. For example, a certain patient may prefer to share medical history only with certain attending physicians and/or nurses, and withhold this information from other staff members.

The challenge is to provide a secure and efficient method for the patient to share his/her previous record(s) based on the desired privacy preferences he/she pleases to maintain. The medical institutions that the patient visits should not struggle to retrieve the record(s) generated previously by other institution(s). This means, records should not be centrally stored and/or controlled by the institution that generates them. In addition to this, the staff members requesting access to the records should only be granted access to specific parts of the record if and only if they possess a specific set of attributes $\mathbb{A} = \{A_1, A_2, \ldots, A_n\}$ predefined by the patient. Staff members that do not possess the correct set of attributes should be denied access in order to honor the privacy desires of the patient.

A. Design Goals

To provide an attribute-based distributed data sharing scheme that is secure and private, we have the following designs goals:

Fine-grained access control: The patient has the capability to encrypt any part of his/her record using any set of descriptive attributes he/she wishes, limiting access to only authorized staff members at particular medical institutions. The set of descriptive attributes is defined by the patient at the time of encryption.

Collusion resistant: Two or more staff members that possess different sets of attributes and/or are employed at the same/different institution(s) cannot combine their attributes to gain access to any part of the record they are not authorized to access individually.

Data confidentiality: All parts of any record are completely protected from any unauthorized staff members that do not possess the correct set of attributes. This includes the storage entity which hosts the records of the patients.

Distributed storage: The records of the patients are stored over a distributed storage space that can be accessed at any time and by any permissioned staff member. The medical institution that generates the record should not control how or whom to share the record with.

B. System Model

The system consists of five main actors and components: patients (data owners), medical institutions (data generators), staff members (data users), key-issuer, IPFS.

Patients $\mathcal{P}$: A set of individuals $\{p_a \in \mathcal{P} \mid 1 \leq a \leq \infty\}$ which receive treatment from different medical institutions.

Medical institutions $\mathcal{M}$: A set of medical institutions $\{m_b \in \mathcal{M} \mid 1 \leq b \leq \infty\}$ that host patients in $\mathcal{P}$ and provide the necessary resources required to treat them.

Staff Members $\mathcal{S}$: The set of staff members $\{s_{b,l} \in m_b \mid 1 \leq l \leq \infty\}$ employed at medical institution m_b that require access to the previous records of the visiting patient.

Key-generator: A semi-trusted entity that generates access keys for eligible staff members which are granted access to certain parts of the record based on the predefined access rights of a staff member.

IPFS: A non-trusted peer-to-peer distributed data storage system used by the patients to store their records they wish to share with staff members at any medical institution.

C. Record Partitioning and Access Rights

We assume that a record $\mathcal{R}$ is partitioned into k parts based on data sensitivity defined by the patient. Each part of $\mathcal{R}$ is encrypted and shared independently in a distributed manner among the staff members of different medical institutions under an attribute-based access structure.

The patient divides the attribute values of his/her record(s) into multiple groups based on how he/she wishes to share it with the staff members. Each group contains attribute

values of similar sensitivity scores defined by the patient. This means, each patient is given the complete freedom to choose his/her privacy preferences. We assume that the patient divides the record into k groups $\{\mathcal{G}_i \mid 1 \leq i \leq k\}$.

D. Group Encryption

The patient generates a set of symmetric keys $\{sk_i \mid 1 \leq i \leq k\}$ corresponding to the previously partitioned groups. The symmetric key sk_1 is randomly selected and used to derive the remaining symmetric keys $\{sk_2, \ldots, sk_k\}$ using a one-way cryptographic hash function h, that is

$$sk_{i+1} = h(sk_i). \tag{1}$$

The patient then encrypts the set of groups using a symmetric encryption algorithm such as the Advanced Encryption Algorith (AES) to generate a set of encrypted groups $\{\mathcal{EG}_i = \mathsf{Enc}_{sk_i}(\mathcal{G}_i) \mid 1 \leq i \leq k\}$.

E. The File System

The patient organizes the encrypted groups into a file system based on the predefined sensitivity. The file system could be a sequence of directories where the most sensitive group $\mathcal{EG}_1$ is placed in the root directory `dir_1` while the least sensitive group $\mathcal{EG}_k$ is placed in the most inner subdirectory `dir_k`. The file system is represented as

```
/dir_1/encrypted_group_1
/dir_1/dir_2
/dir_1/dir_2/encrypted_group_2
/dir_1/dir_2/dir_3
                    ⋮
/dir_1/.../dir_k/encrypted_group_k
```

Once the file system is created, the patient uploads it to IPFS by running the following command from the root directory.

```
$ ipfs add -r .
```

IPFS will partition the uploaded file system accordingly, distribute the partitions in the network nodes, and update the DHT to reference the uploaded data. The patient may use the reference links stored in the DHT to access the encrypted data and share it with staff members upon making future visits.

F. Attribute-Based Encryption

After the patient encrypts the groups and uploads the file system to IPFS, the symmetric keys are encrypted under the attribute-based encryption scheme [10] to be accessed only by the staff members that possess the correct set of attributes. Our proposed scheme follows the construction presented in [10] and formally divides the process into four main functions: $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$.

$\mathsf{Setup}(1^\kappa)$*:* The patient may delegate performing this probabilistic function to a semi-trusted key-issuer. The Setup function takes a security parameter κ and randomly chooses values $\alpha, \beta \in \mathbb{Z}_p$. The outputs of this function are public key PK and master key MK defined as

$$PK = \{\mathbb{G}_0, g, B = g^\beta, e(g,g)^\alpha\}, \tag{2}$$

$$MK = \{\beta, g^\alpha\}. \tag{3}$$

$\mathsf{KeyGen}(MK, \mathbb{A})$*:* This is a probabilistic function also carried out by the key-issuer. The inputs to this function are MK generated by the Setup function and an attribute set $\mathbb{A}$ possessed by the staff member $s_{b,l}$, where $A_u \in \mathbb{A}$ represents the u^{th} attribute within the set. The KeyGen function outputs a unique private key SK for the staff member. In order to guarantee a unique SK, it generates a random value $r \in \mathbb{Z}_p$ and incorporates it into the private key. Based on the number of attributes in the input set $\mathbb{A}$, the KeyGen function also generates a random value $r_u \in_r \mathbb{Z}_p$ for each attribute within the set. The SK is defined as:

$$\begin{aligned} SK = \big(&D = g^{(\alpha+r)/\beta}, \\ &\{D_u = g^r \cdot h(u)^{r_u}, D'_u = g^{r_u} \mid \forall A_u \in \mathbb{A}\}\big) \end{aligned} \tag{4}$$

$\mathsf{Encrypt}(PK, sk_i, \mathcal{T}_i)$*:* This is a probabilistic function carried out by the patient for each symmetric key to be shared. The inputs to this function are PK, the public key generated by the Setup function, the symmetric key sk_i to be encrypted, and the access tree $\mathcal{T}_i$ that defines the authorized set of attributes. Patients define $\mathcal{T}_i$ in the way that they please to maintain the privacy of their records. The output of this function is the ciphertext CT_i that will be shared with permissioned staff members.

The $\mathsf{Encrypt}$ function chooses a polynomial $q_{x_l^i}$ with degree $d_{x_l^i} = k_{x_l^i} - 1$ for each node $x_l^i \in \mathcal{T}_i$. The process of assigning polynomials to each x_l^i occurs in a top-bottom approach starting from the root node in $\mathcal{T}_i$. The $\mathsf{Encrypt}$ function chooses a secret $s \in \mathbb{Z}_p$ and sets the value of $q_{x_1^i}(0) = s$. Next, it randomly chooses the remaining points of the polynomial to completely define it. For any other node $x_l^i \in \mathcal{T}_i$, the $\mathsf{Encrypt}$ function sets the value $q_{x_l^i}(0) = q_{\text{parent}}(\text{index}(x_l^i))$, where q_{parent} is x_l^i's parent node polynomial. The remaining points of those polynomials are then randomly chosen.

Let X_i be the set of leaf nodes in $\mathcal{T}_i$. The CT_i is then constructed as

$$\begin{aligned} CT_i = \big(&\mathcal{T}_i, \tilde{C}_i = sk_i \cdot e(g,g)^{\alpha \cdot s}, C_i = g^{\beta \cdot s}, \\ &\{C_{x_l^i} = g^{q_{x_l^i}(0)}, C'_{x_l^i} = h(x_l^i)^{q_{x_l^i}(0)} \mid \forall x_l^i \in X_i\}\big). \end{aligned} \tag{5}$$

$\mathsf{Decrypt}(CT_i, SK)$*:* This is a deterministic function carried out by the staff member. The inputs to this function are the ciphertext CT_i generated by the $\mathsf{Encrypt}$ function and the private key SK of the staff member $s_{b,l}$.

The $\mathsf{Decrypt}$ function operates in a recursive manner propagating through the nodes in $\mathcal{T}_i$ by calling a recursive function defined as $\mathsf{DecryptNode}(CT_i, SK, x_l^i)$. The inputs to this function are CT_i, SK and a node x_l^i within $\mathcal{T}_i$. If the attributes incorporated within the private key of the staff member satisfy the nodes within $\mathcal{T}_i$, the staff member can decrypt CT_i.

$\mathsf{DecryptNode}(CT_i, SK, x_l^i)$ performs differently depending on whether x_l^i is a leaf or non-leaf node. If x_l^i is a leaf node and $\text{att}(x_l^i) \notin \mathbb{A}$, $\mathsf{DecryptNode}(CT_i, SK, x_l^i)$

returns $\emptyset$, otherwise, $\text{att}(x_l^i) = A_u \in \mathbb{A}$ and $\mathsf{DecryptNode}(CT_i, SK, x_l^i)$ is defined as:

$$\mathsf{DecryptNode}(CT_i, SK, x_l^i) = \frac{e(D_u, C_{x_l^i})}{e(D'_u, C'_{x_l^i})} = e(g,g)^{r \cdot q_{x_l^i}(0)}. \tag{6}$$

If x_l^i is a non-leaf node, $\mathsf{DecryptNode}(CT_i, SK, x_l^i)$ operates recursively. For each node $z_{l,c}^i$ that is a child of x_l^i, $\mathsf{DecryptNode}(CT_i, SK, z_{l,c}^i)$ is computed and the output is stored in $F_{z_{l,c}^i}$.

This recursive function is based on Lagrange interpolation. The Lagrange coefficient $\Delta_{a,\mathbb{A}}$ for $a \in \mathbb{Z}_p$ and the set of attributes $\mathbb{A}$ is defined as:

$$\Delta_{a,\mathbb{A}}(x) = \prod_{k \in \mathbb{A}, k \neq a} \frac{x-k}{a-k}. \tag{7}$$

Let $\mathbb{A}_{x_l^i}$ be an arbitrary $k_{x_l^i}$-sized set of child nodes $z_{l,c}^i$ such that $F_{z_{l,c}^i} \neq \emptyset$. If no such set exists then the function returns $F_{z_{l,c}^i} = \emptyset$. Otherwise, $F_{z_{l,c}^i}$ is computed using Lagrange interpolation as follows:

$$F_{x_l^i} = \prod_{z_{l,c}^i \in \mathbb{A}_{x_l^i}} F_{z_{l,c}^i}^{\Delta_{a,\mathbb{A}'_{x_l^i}}(0)} = e(g,g)^{r \cdot q_{x_l^i}(0)}, \tag{8}$$

where $a = \text{index}(\text{z}_{\text{l,c}}^{\text{i}})$ and $\mathbb{A}'_{x_l^i} = \{\text{index}(z_{l,c}^i) \colon z_{l,c}^i \in \mathbb{A}_{x_l^i}\}$.

If the attributes in $\mathbb{A}$ satisfy $\mathcal{T}_i$, then the following is computed at the root node x_1^i.

$$\begin{aligned} R_i &= \mathsf{DecryptNode}(CT_i, SK, x_1^i) \\ &= e(g,g)^{r \cdot q_{x_1^i}(0)} = e(g,g)^{r \cdot s}. \end{aligned} \tag{9}$$

To obtain sk_i from the result derived in equation (9), we compute the following:

$$\frac{\tilde{C}_i}{\frac{e(C_i, D_j)}{R_i}} = \frac{sk_i \cdot e(g,g)^{\alpha \cdot s}}{\frac{e\left(g^{\beta \cdot s}, g^{(\alpha+r)/\beta}\right)}{e(g,g)^{r \cdot s}}} = sk_i. \tag{10}$$

The staff member may also derive the symmetric keys $\{sk_{i+1}, \ldots sk_k\}$ as discussed in equation (1).

G. Data Retrieval

At this point, the staff members with the correct set of attributes will possess the symmetric keys $\{sk_i \ldots sk_k\}$ corresponding to certain partitions of the record. In order for staff members to retrieve the data, the patient must provide the staff members with the IPFS storage locations of the granted groups. Once provided, the staff members can retrieve these using the command

```
$ ipfs get block_address
```

where block_address is the address of the root partition of the shared group. The nodes in the network can then find the blocks linked to this address using the DHT and download the entire encrypted group. Using block_address, the staff members can also find the location of the groups with less sensitivity using the DHT. Therefore, the staff members can retrieve the encrypted groups $\{\mathcal{EG}_i, \ldots, \mathcal{EG}_k\}$ based on the access tree $\mathcal{T}_i$ satisfied. Finally, using the symmetric keys discussed in section III-F, the staff members can decrypt each encrypted group using their corresponding keys to obtain the information of the patients.

IV. Security Analysis

In this section, the security of our proposed model is analyzed.

A. Two-Layer Security

Our proposed scheme provides two separate layers of security to the stored records. The first layer is provided through the symmetric encryption of the data performed by the patient before uploading the record to IPFS. We assume a symmetric encryption scheme such as AES is used when encrypting the records of patients. Staff members must possess a specific set of attributes that satisfy an access policy corresponding to the record parts they wish to retrieve. Staff members in possession of these attributes are able to request the corresponding symmetric keys from the key generator that allow them to decrypt the ciphertext once retrieved from IPFS. The second layer of security is provided by the distributed storage network. Data files are partitioned, cryptographically-hashed, and distributed over the peer-to-peer network. Retrieving the data from IPFS requires knowing its exact storage address maintained by the DHT. However, for the current version of IPFS, the first partition always carries the syntax of the original file. Therefore, the extra layer of encryption is essential to ensure file confidentiality.

B. Content-Addressable Storage

Data stored over IPFS is content-addressable. The cryptographic-hash of data is stored in the DHT and utilized by nodes to locate its storage location. A user in possession of the data will be able to reproduce its cryptographic-hash, search for its corresponding location maintained by the DHT, and locate it within the peer-to-peer network nodes. This will also result in recursively locating any data linked to it.

Consider data files that have not been encrypted before being uploaded to IPFS to enhance performance. A valid argument might be that an attacker may exhaustively compute the cryptographic-hash of different values of a record, assuming the stored data was partitioned with the default IPFS partitioning scheme and the attributes of a record are known and fixed. Therefore, the attacker might be able to locate the stored data over IPFS and obtain all the data linked to it. However, a simple solution to this issue would be for the patient to append a random value r to the record before uploading it to IPFS. This will result in a different digest $h(\mathcal{R}) \neq h(\mathcal{R} + r)$ when cryptographically-hashing the record and makes it infeasible for the attacker to locate the data stored over IPFS. In addition to this, IPFS by default partitions data into blocks of 256KB then applies SHA2-256 hash function to each partition. Changing these parameters will result in different digests that are stored in the DHT used to locate the data in IPFS.

C. Semi-Trusted Key Generator

Due to the nature of our two-layer security proposed scheme, the key generator is no longer required to be a completely trusted entity as with centralized schemes. A patient only requires to trust a key generator that will generate access keys for the staff members with the correct set of attributes upon their requests. A key generator that maliciously generates access keys for staff members that do not possess the correct set of attributes will not be able to retrieve any data without being able to locate it in the DHT.

V. Performance Evaluation

To evaluate the performance of our proposed scheme, simulations have been conducted to measure the time it takes to write and read records to the distributed storage IPFS and the centralized storage Google drive. The results of the simulations are shown in Fig. 1 and 2. In the simulations, we write and read records that consist of nine attributes. We consider three scenarios with different record sizes, 256KB, 5MB, and 15MB. We also assume three different levels of sensitivity $k = 3$ where a record is divided into three groups. The first group $\mathcal{G}_1$ with the highest sensitivity contains two attributes, the second group $\mathcal{G}_2$ contains three attributes, and the third group $\mathcal{G}_3$ with the least sensitivity contains four attributes.

As shown in Fig. 1, the time required to write records of different sizes to IPFS outperforms writing them to Google drive. As the size of the record increases, the writing time to IPFS increases slightly while for Google drive, the time increases linearly. This is attributed to dividing the records into partitions before allocating them to the network nodes to be stored.

In Fig. 2, we demonstrate the simulation results for reading the stored records that have been written previously. We test the retrieval time for each record from both IPFS and Google drive. As shown in the figure, the second retrieval time for both IPFS and Google drive decreases. However, the performance of reading records stored over IPFS surpasses that of reading the records stored over Google drive in both reads. The peers of the requesting node can easily locate the requested data and/or keep a local copy to supply the node with the data upon its second request.

It is also worth mentioning that IPFS does not increase the file size when storing the data in the peer-to-peer network.

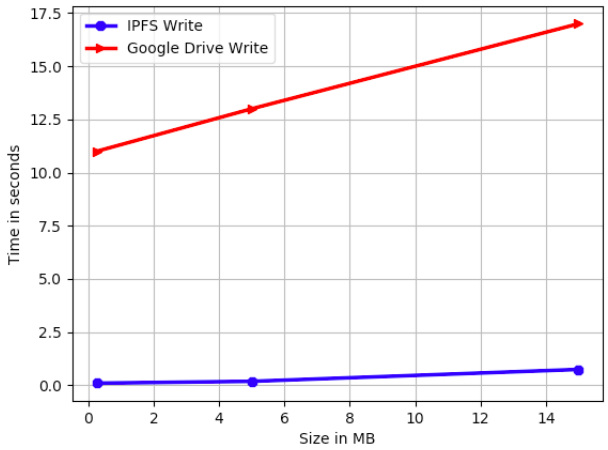

Fig. 1: Write time.

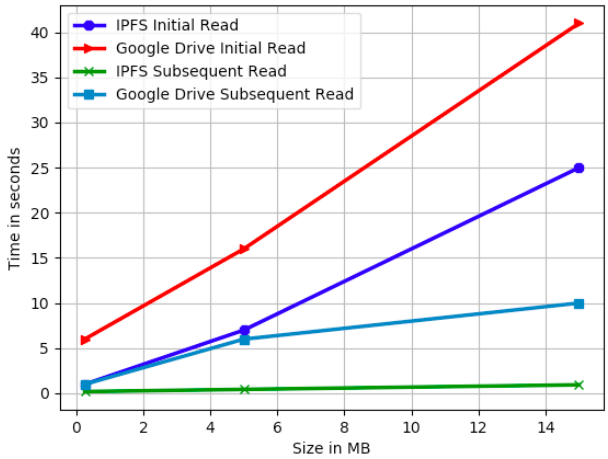

Fig. 2: Read time.

VI. Conclusion

In this paper, we present an attribute-based distributed data sharing scheme. Our proposed scheme enables patients to control where their records are stored and how they are shared instead of having to trust medical institutions. We showed that our proposed scheme can provide a certain level of privacy protection in addition to record confidentiality. The security analysis proves that our proposed scheme enhances security and does not require fully trusted entities. We also proved that storing the records over distributed storage enhances the availability of the records upon their request. Our simulation results also demonstrated that our proposed scheme requires less writing and reading access time than the centralized storage.

References

[1] USA.gov, "Health information exchange." http://www.healthit.gov/topic/health-it-basics/health-information-exchange, 2017.

[2] aph.gov.au, "Patient access to medical record." https://www.aph.gov.au/Parliamentary_Business/Committees/House_of_Representatives_Committees?url=laca/privacybill/chap7.pdf.

[3] J. Benet, "Ipfs - content addressed, versioned, p2p file system (draft 3)." https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf.

[4] A. Beimel and A. Schimmel, *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[5] I. Baumgart and S. Mies, "S/kademlia: A practicable approach towards secure key-based routing," in *Parallel and Distributed Systems, 2007 International Conference on*, pp. 1–8, IEEE, 2007.

[6] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing content publication with coral.," in *NSDI*, vol. 4, pp. 18–18, 2004.

[7] "Bittorrent." http://www.bittorrent.com, year = 2018.

[8] T. Cipriani, "Visualizing git's merkle dag with d3.js." https://tylercipriani.com/blog/2016/03/21/Visualizing-Git-Merkle-DAG-with-D3.js/, year = 2016.

[9] "Git." https://git-scm.com/docs, year = 2018.

[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334, IEEE, 2007.