Date of publication xxxx 00, 0000, date of current version March 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.01640

HealthBlock: A Modular Framework for a Collaborative Sharing of Electronic Health Records Based on Blockchain

LEINA NAZAR ABDELGALIL [1] **MOHAMED MEJRI[2]**
[1]Sudan University of Science and Technology, Khartoum, Sudan (e-mail: leina.nazar@gmail.com)
[2]Laval University, Quebec, Canada (e-mail:mohamed.mejri@ift.ulaval.ca)

ABSTRACT Electronic Health Records (EHRs) play an important role in our life. However, most of the time, they are scattered and saved on different databases belonging to distinct institutions (hospitals, laboratories, clinics, etc.) geographically distributed across one or many countries. Due to this decentralization and the heterogeneity of the different involved systems, the medical staff are facing difficulties in correctly collaborating by sharing, protecting, and tracking their patient's electronic health-record history to provide them the best cares. Also, patients have no control on their private EHRs. Blockchain has many promising futures useful for the healthcare domain because it provides a better solution for sharing data while preserving the integrity, the interoperability, the availability, and even the privacy than classical client-server architectures used to manage EHRS.

This paper proposes a modular framework called HealthBlock for a collaborative sharing EHRs and their privacy-preserving with a granular access control. Different technologies have been combined to access this goal: the InterPlanetary File System (IPFS) technology stores and shares patients' EHRs in distributed Off-Chain storage and ensures the record's immutability; the Hyper-ledger Indy gives patients full control over their EHRs and the Hyper-ledger Fabric store the patient access control policy and delegations.

INDEX TERMS Electronic Health Records (EHRs), Blockchain, Hyperledger Fabric, Hyperledger Indy, Self-Sovereign Identity (SSI), Smart Contracts, InterPlanetary File System (IPFS).

I. INTRODUCTION

Most existing healthcare organizations use local databases to hold the EHRs of their patients. However, even if they do their best to protect their clients' private information, this solution suffers many drawbacks. In fact, during their lifetime, patients usually request different healthcare-service providers, and their EHRs will be scattered, and critical information might not be available anywhere at vital moments. This scenario is frequent even if the patients' healthcare providers are in the same city. The situation will be worst for people that visit healthcare providers in different countries.

The patients themselves have not always access all their records, and they have no control over who can access what. As a result, sensitive information is sometimes leaked, causing damage to their owners: losing a job, increased insurance fees, etc.

Several new building blocks (blockchains, hyperledgers, IPFS, etc.) are now available to develop secure architectures for different applications more easily and even with new highly requested features. Classical cryptographic systems (encryption, signature, hash, etc.) provide basic blocks to easily achieve certain security properties such as confidentiality, integrity and authentication. However, other objectives (availability, transparency, immutability, automation, etc.) require, in addition to the combination of several classic cryptographic systems, the integration of other concepts (peer-to-peer networks, consensus algorithms, smart contracts, etc.). IPFS [1], for example, uses the Bitswap peer-to-peer network (a generalization of the BitTorrent protocol) [2], the GIT version management system [3], a self-certifying File System (SFS) [4] and the hash tree (Merkle tree) to distribute files across multiple nodes to avoid a single point of failure or the need of trusted nodes. The hyperledgers and blockchains use peer-to-peer networks, consensus algorithms, smart contracts, signature, hash functions, ZKP protocols, etc., to achieve interesting properties such as availability, integrity, transparency, traceability and automation. Using smart con-

tracts for instance, we can automatically trigger, under certain conditions, some processing without the intervention of any trusted authority. The contract is executed even if the expected results are against the will of the stakeholders.

These new emerging technologies, including blockchains, provide a better solution for patients' record protection, accessibility, delegation, anonymity and other useful features.

The main goal of this paper is to address the problems of the existing EHRs management systems by providing a solution with interesting features such as:

- the EHRs are appropriately protected (Integrity + Confidentiality) and available anytime from any country.
- the patients have full control of their EHRs, and they can temporarily delegate access to any person or organization at any time.
- the patients can get anonymous healthcare services, when needed, to better protect their privacy.
- the patients' attributes can be proved remotely, making telemedicine more efficient.
- the solution takes into consideration that in an emergency, the patients may not have their full capacity to grant access to their EHRs.

Different technology, including the hyperledger Fabric, the hyperledger Indy, and the IPFS database are combined to access these goals.

The remaining part of this paper is organized as follows. Section II recalls some preliminary notions useful to understand the proposed architecture. Section III presents the related work. Section IV gives the proposed framework architecture for sharing EHRs. Section V provides a prototype as a proof of concepts. Section VI discuss the proposed approach and Section VII provides some concluding remarks.

II. PRELIMINARIES

We start by briefly introducing the principal technologies involved in our architecture.

A. BLOCKCHAIN TECHNOLOGY

A Blockchain is a shared, immutable, distributed ledger of cryptographically signed transactions grouped into blocks. Each block is linked to the previous one by including its hash. Many nodes have the exact copy of the blocks and collaborate according to a consensus algorithm to add new transactions that respect predefined criteria.

1) Key Characteristics of Blockchains

- **Ledger**: Unlike traditional databases, transactions in a Blockchain could not be modified or deleted.
- **Secure**: Blockchains intrinsically provide integrity, traceability, and availability.
- **Absence of trust**: A Blockchain can fulfill its security properties even if its nodes are operated by players that do not trust each other and have antagonist goals.

2) Blockchain Types

There are different types of Blockchains and Hyperledgers. They could be public or private and with or without permission. For instance, Hyperledger can be accessible to everyone (public) but with different permission (some users can read, others can read and write). Another could limit the access to a reduced group (private) where each group member can have different permission (private, with permission). Different examples of public and private Blockchains/Hyperledgers, with and without permission, are given in Table 1.

	Without permission read and write	**With permission** limited read or write
Public (access to everyone)	Bitcoin, Ethereum, IOATA, etc.	Hyperledger Indy (sovereign), IODB, etc.
Private (access to a limited group)	Hyperledger Sawtooth (permissionless mode)	Hyperledger Fabric, Iroha, Sawtooth, etc.

TABLE 1. Blockchain Types.

B. SMART CONTRACTS

Smart contracts are self-executing computer programs deployed on the Blockchain to generate new facts added to the ledger. They are utilized in different fields, such as healthcare, financial services, and government. They are also called *Chaincode* in hyperledger Fabric and written using different programming languages such as Solidity, JavaScript, Go, Java and Rholang. Once it is deployed in the Blockchain, a smart contract cannot be changed or removed. Usually, it simply waits for external events to execute some specific task and update the system's state.

C. HYPERLEDGER FABRIC

The Linux Foundation created the Hyperledger Fabric [5] in 2015. It aims to be a general-purpose Hyperleder that satisfies a broad range of applications. It is an open-source, modular, and permissioned Hyperledger that offers a granular access control.

D. INTERPLANETARY FILE SYSTEM (IPFS)

The InterPlanetary File System (IPFS) [1] is an immutable and distributed file system for storing and sharing data. It works using a peer-to-peer (P2P) network. Files are stored in IPFS objects, each object can contain up to 256Kb of data as well as links to other objects. Big files are split into a multiple objects, and a new object with an empty data containing the links to all slices is also created. IPFS provides a content-addressing technique to uniquely identify a file: the address of any file is simply its hash value. As in a Blockchain, nodes do not need to trust each other and the files remain available even if many nodes are disconnected. Files inserted in IPFS cannot be changed anymore, ensuring immutability, but it supports versioning. For every version of a file, a commit object is created and linked to each other to easily access to the history of each file.

III. RELATED WORK

We have studied many research papers related to EHRs management systems during the last few years. We can evaluate them according to some useful criteria, including the following:

- **Patients have full control over their EHRs:** Patients keep their EHRs in their wallets or saved in some store in an encrypted form, and they control who can access to selected attributes and when.
- **Anonymous Healthcare Service:** It will be nice if the solution gives patients the possibility to get an anonymous service when they want. Patients hold a set of attributes related to their identity, including EHRs, and show only what it is required. For instance, to get a Covid-19 vaccine, patients may need only to show that they are older than a certain age; they are citizens of the city where they ask for the vaccine and have not received the vaccine yet.
- **Availability:** It is very important that the EHRs will be available anytime from any location, even if some equipment is down or attacked.
- **Integrity:** We should ensure that EHRs have not been accidentally or maliciously changed. Without integrity, EHRs are useless.
- **Confidentiality:** EHRs contain very sensitive information, and if disclosed, some serious damages can take place for their owners, including loss of jobs and increasing insurance fees.
- **Access Control Delegation:** Any solution that aims to manage EHRs should give the patient an easy way to delegate some access to any attributes related to their EHRs.
- **Zero-Knowledge Proof (ZKP):** It could be useful that patients can prove some properties related to their attributes without revealing their values. For example, they can prove that they are more than 18 years old without revealing their exact age. Commonly, some Blockchains used ZKP protocols to provide this kind of property.
- **Emergency Access to EHRs:** In case of an emergency when the patient is not in his full capacity to provide by himself, his EHRs, a secure alternative solution should be provided.
- **Scalability:** When Blockchains are used, we should carefully evaluate the volume of data added every day and be sure that the solution stays scalable even when billions of clients are involved.
- **Interoperability:** To maximize its interoperability, any solution should use standards and, when needed, Blockchain should be public and available worldwide.
- **Remote Health Service:** EHRs management solutions should provide the possibility of remote consultations, a growing need, particularly in the covid-19 context. This could be possible, particularly when patients could remotely prove their attributes.

Hereafter, we shortly describe the main interesting EHRs management solutions that we have included in our comparative studies.

Sun et al. proposed in [7] a scheme to efficiently and securely store electronic medical data. They used attribute-based encryption techniques combined with Blockchain technology to control electronic medical data access. The EHRs are encrypted and stored in the decentralized InterPlanetary File System(IPFS) storage, providing scalability, privacy and preventing a single point of failure.

Kumar et al. proposed in [8] a distributed off-chain storage to manage patients' medical data. They store the medical records in the IPFS and their indexes in the Blockchain. The approach aims to provide consistency, integrity, and availability for medical data.

In [9], Khatoo proposed an Ethereum-based solution to manage healthcare records, where the social security number is used to map the users' Ethereum address. The author uses smart contracts to provide flexibility, interoperability, and accessibility of patients' medical data.

Nguyen et al. proposed in [12] an EHRs sharing framework on a mobile-cloud platform. They used Ethereum Blockchain technology hosted on Amazon Cloud to implement a health data sharing framework on mobile clouds and protect patients' sensitive information from malicious attacks. Also, they profit from the smart contracts on Ethereum to manage access control on these EHRs.

In [10], Shen et al. provided a Blockchain-based solution called MedChain to share medical data efficiently. $MedChain$ combines Blockchain technology, digest chains, and structured peer-to-peer network techniques to overcome the efficiency issues in the existing solutions for sharing healthcare data. $MedChain$ consists of two different peer nodes, namely super peers (such as national hospitals having servers with high computing and storage capacities) and edge peers (such as community clinics that store local patients' data).

In [11], Niu et al. suggested a secure EHRs sharing schema based on permissioned Blockchains. The approach supports multiuser search and uses the ciphertext-based attribute encryption mechanism to achieve data confidentiality, provides a fine-grained access control, and prevents malicious doctors from uploading false information through an authenticated access.

Shahnaz et al. proposed in [13] an Ethereum-based Blockchain architecture to handle EMRs. They used Solidity to write smart contracts that control access to healthcare records.

Kim et al. suggested in [14] a data-sharing medical questionnaire management system based on Blockchain technology. They leveraged Blockchain characteristics to guarantee the integrity of data. They also address the scenario when a third party requests some EHRs attributes requiring the patient's permission.

In [16], Dagher et al. leveraged Ethereum Blockchain technology features and proposed a framework called $Ancile$ to

secure and appropriately control access to EHRs that should be made available to various users (patients, providers, and third party) under different conditions.

Fan et al. suggested in [17] a framework called $MedBlock$ based on Blockchain technology to manage access control on EHRs. Using $MedBlock$, the patients can easily share and access their EHRs with the minimum energy consumption.

In [18], Liu et al. proposed a scheme for Electronic Medical records (EMRs) sharing and privacy-preserving based on Blockchain technology called BPDS. In BPDS, the original EMRs are stored securely in a cloud storage under the management of smart contracts, and their indexes are stored in a consortium Blockchain to ensure that EMRs cannot be falsified.

In [19], Al Omar et al. provided a privacy-preserving medical data platform based on a Blockchain where medical data are encrypted and stored.

In [20], Dubovitskaya et al. proposed a Blockchain-based framework for sharing EHRs for cancer patients. They leveraged permissioned Blockchain features to achieve immutable, accountable, and fine-grained EHR access control. In this framework, the EHRs are encrypted using the patients' public keys and stored in a cloud server. When the data is needed, it is necessary to obtain the data owner's decryption key.

Liang et al. suggested in [21] a model for user-centric sharing data in e-health systems. They leveraged a permissioned Blockchain to protect the healthcare data privacy and enhance the decentralized identity management using membership services supported by Blockchain. In this framework, the healthcare data are collected from different sources, including wearable devices, and stored in the cloud for sharing purposes between different providers and insurance companies.

Xia et al. [22] proposed $MeDShare$, an Electronic Medical Records (EMRs) manager based on a Blockchain, and provide access control for medical data in a trustless environment. EHRs are stored in the cloud, and access control is achieved through smart contracts.

In [24], Azaria et al. proposed MedRec medical data access and permission management system based on a permissionless Blockchain technology. Also, this framework provides health data sharing between healthcare systems. The use of the Blockchain technology provides accountability, confidentiality, traceability, and sharing of EMRs. They designed three Ethereum smart contracts to achieve fine-grained access control for patients' medical records. The Blockchain does not store actual medical data; it remains in the healthcare system's database.

Yue et al. [25] proposed a Blockchain-based storage platform called Health Data Gateway(HDG) that allows patients to control and share their medical data without violating their privacy. The patient's medical data is encrypted by the providers and stored in a private Blockchain. When needed, the patient downloads his encrypted medical data, decrypts it using his private key, and decides whether to share it or not.

Many other eHelth solution such as [26], [27] hve been proposed even before the appearance of the blockchain. They are often based on cryptographic primitives and crytoprotocols to ensure many security properties such as the confidentiality and the integrity of EHRs.

Table 2 resume the different connection between the presented related work according to the used technologies.

Table 3 evaluates the above works according to the previous criteria, including Anonymous Healthcare Service, Availability, Integrity, Confidentiality, Access Control Delegation, Zero-Knowledge Proof(ZKP), Emergency Access to EHRs, Scalability, Interoperability, Remote Health Service.

IV. PROPOSED FRAMEWORK

The proposed framework is presented incrementally. We first present a core solution based on the hyper-ledger Indy, discussing its features. After that, we propose different extensions based on other exciting technologies such as the hyperledger Fabric [5], the IPFS [1], and Fido U2F [28].

A. PRELIMINARIES

1) Roles

Three prominent roles are involved in this framework of a Collaborative Sharing of EHRs Framework Based on Blockchain Technology:

1) **Patients**:
   Patients play a crucial role, and the whole system aims to appropriately protect their EHRs and make them available to authorized participants when needed. Each document (a part of an EHRs, insurance, identity, etc.) contains a set of attributes (name, address, blood type, etc.) and a unique DID (Decentralize Identifier). Each DID is also associated with a public key and stored in the Indy hyperledger. A patient proves that he/she is the owner of a document by showing he/she is the owner of the private key associated with the public correlated to the document referred by DID. The patients have full control of their EHRs. The system allows them to create and modify the access control policies related to their EHRs. This policy could be stored in a hyperledger Fabric or Ethereum. The EHRs are generally scattered and distributed in different healthcare service providers. To simplify the presentation, we assume that Bob is our patient.
2) **Healthcare providers**: They are the producers and the consumers of healthcare attributes related to patients. They include individuals (doctors, nurses, etc.) and institutions (hospitals, labs, research centers, etc.). Once a healthcare attribute is produced or updated, any future access, inside or outside the same institution, should be consistent with the patient access control policy. To simplify the framework's presentation, we assume that the provider is a laboratory L and the doctor Alice is the consumer.
3) **Healthcare card provider or insurance issuers**: Using their healthcare cards or insurance, patients

Blockchain Type	Blockchain Name	Smart Contract	Blockchain Role	EHRs Access Control	EHRs Privacy	EHRs Storage	Approach
Public	Ethereum	Yes	Save Address of EHRs Save Addresses of Patients	Attribute Based Encryption (ABE) Cloud EHR Manager	ABE EHR Manager's Public Key	IPFS	Sun et al. 7 Nguyennet al. 12
						Off-chain Storage	Shahnaz et al. 13 Dagher et al. 16
						Provider Local Database + Patient Database	Azaria et al. 24
	Not fixed	No	Save hash value of EHRs Store data digest in Blockchain	bread crumbs	Encryption technology	Local Database or Cloud Database	Fan et al. 17 Shenn et al. 10
Consortium	Not fixed	No	Save keywords of EHRs	Attribute-Based Encryption	Encryption technology	Cloud Storage	Niu et al. 11 Liang et al. 21
			Save hash of medical data	membership service component			
			Save medical data	Not specified		Blockchain	Al Omar et al. 19
			metadata+access control policy	Role-based		Local Database + Cloud	Dubovitskaya et al. 20
			Stored hash of medical records	Not specified	Hash based data storage	IPFS	Kumaret al. 8
		Yes	Reserve indexes of EMRs	Content extraction signature	Encryption technology	Cloud Blockchain	Liu et al. 18 Kim et al. 14
			Questionnaire result data	According to medical policy			
			save immutable database	Not specified		Database	Xia et al. 22
Private	Not Fixed	No	data stored in private blockchain	Access control model	Encryption technology	Blockchain + Cloud	Yue et al. [25]
No Blockchain	No Blockchain	No	No Blockchain	Access control	Encryption technology + Cryptographic Protocols	Database	Sandıkkaya et al. [26]

TABLE 2. Commonalities of Related Work

Author	Patient have full control over their EHRs	Anonymous Healthcare Service	EHR-Availability	EHR-Confidentiality	EHR-Integrity	EHR-Access Control Delegation	EHR-Zero-Knowledge Proof	EHR-Emergency Access	Scalability	Interoperability	Remote Health Service
Sun et al. 7			✓	✓	✓						
Kumarnet al. 8			✓	✓	✓						
Khatoo 9			+-	+-	+-						
Nguyennet al. 12			✓	✓	✓						
Shenn et al. 10			+-	+-	✓						
Niu et al. 11			+-	✓	✓						
Shahnaz et al. 13			✓	✓	✓				✓		
Kim et al. 14			✓	✓	✓						
Dagher et al. 16			+-	+-	✓					✓	
Liu et al. 18			+-	+-	✓						
Al Omar et al. 19			✓	✓	✓						
Dubovitskaya et al. 20			✓	+-	+-						
Liang et al. 21			+-	+-	✓				✓		
Xia et al. 22			+-	+-	✓						
Azaria et al. 24			+-	✓	✓					✓	
Yue et al. [25]			✓	✓	✓						
Fan et al. 17			+-	+-	+-						

- ✓means that the approach given in the line has the feature given in the corresponding column.
- +- means that this feature is not always present, it depends on the provider-side security level.
- An empty case means that the proposed approach has not the feature given in the corresponding column.

TABLE 3. A Comparison Between Related Work

can receive services from different healthcare providers without directly paying them when they are appropriately covered.

2) Notations

In the remaining part of this paper, we adopt the following notation related to the cryptographic operations.

- $H(m)$ (Hashed message): A message m hashed by a hash function H such as SHA-256, SHA3, RIPMD, Merkle-tree, etc.
- $\{m\}_k$ (Symmetric-key encryption): A message m encrypted by the key k using a symmetric cryptographic system such as AES, 3-DES, etc.
- $\{m\}_{k_a}$ (Message encrypted with a public key): A message m encrypted with the public key k_a using an asymmetric cryptographic system such as RSA, El-Gamal, etc.
- $\{m\}_{k_a^{-1}}$ (A signed message): A message m signed with the private key k_a^{-1} using an asymmetric cryptographic system such as RSA, DSA, El-Gamal, etc.

3) Indy Hyperledger

It is a permissioned public hyperledger where anyone can read from the ledger, but only authorized principals can write on it. It is based on Sovrin [29] and aims to offer a self-sovereign identity based on a Blockchain. For example, the government of British Columbia uses it in Canada to implement its eID solution [30].

It could be seen as a ledger that contains certificates linking DIDs to public keys and signed by a Trust Anchor `TA`. A `TA` is an individual or an organization (government, insurance, hospital, clinic, etc.) that is already known and trusted by the Indy hyperledger. They act as certification authorities, adding a new Trust Anchor to the ledger. They have the right to write in the ledger. Whenever a `TA` delivers a document (an Identity) containing fresh DID, they add a link (certificate) between the DID and its public key to Indy, as shown in Table 4. This entry is added to the ledger to attest that the owner of the identity containing the DID D_a is the one who knows the private key associated with K_a. The main information of this entry are "D_a, K_a^0, DID-`TA`, $\{H(D_a, K_a, \text{DID-TA})\}_{K_{\text{TA}}^{-1}}$", where DID-`TA` is the DID of the Trust Anchor that has signed this record. Indy contains many transactions having this

kind of record. The same TA could sign many associations between DIDs and public keys, as shown in Table 4.

DID	PUBLIC-KEY	DID-TA	$\{Hash(\text{DID}, \text{PUBLIC-KEY}, \text{DID-TA})\}_{K_{TA}^{-1}}$
D_a^0	K_a^0	D_{TA_1}	$\{H(D_a^0, K_a^0, D_{\text{TA}_1})\}_{K_{\text{TA}_1}^{-1}}$
D_a^1	K_a^1	D_{TA_1}	$\{H(D_a^1, K_a^1, D_{\text{TA}_1})\}_{K_{\text{TA}_1}^{-1}}$
D_b	K_b	D_{TA_2}	$\{H(D_b, K_b, D_{\text{TA}_2})\}_{K_{\text{TA}_2}^{-1}}$
D_c	K_c	D_{TA_3}	$\{H(D_c, K_c, D_{\text{TA}_2})\}_{K_{\text{TA}_3}^{-1}}$
⋮	⋮	⋮	⋮

TABLE 4. Links Between DIDs and Public Keys in the Indy Hyperledger

The Hyperledger Indy also contains information related to TAs. We find a certificate for each trusted authority containing its DID, its service endpoint (URI or IP address, transport protocol, and port), public key, the schema of its delivered documents (the list of attributes that could be found in the credential), etc. Therefore, having the DID of a document, we can easily get the producer's public key and join him through its service endpoint.

Any part of an EHRs could be considered a credential containing a claims list of pairs(attribute, value). The credential also contains the proof of claims involving the issuer's signature, as shown in Figure 1.

Credential		
Metadata	:	issuer, expiration date, etc.
claim(s)	:	$attribute_1 : value_1, \ldots, attribute_n : value_n$
proof(s)	:	signature information

FIGURE 1. Credential Format in Indy Hyperledger

Assume that an agent B wants to prove to A that he has an attribute included in a credential containing the DID D_b; then, they need to proceed as follows:

- A reads from the hyperledger Indy, the public key k_b associated with the DID D_b.
- A asks B to prove that he owns the private key associated with k_b using a challenge-response protocol such as:

$$\begin{array}{llllll} 1. & A & \rightarrow & B & : & A, B, N_a \\ 2. & B & \rightarrow & A & : & \{H(A, B, N_a)\}_{k_b^{-1}} \end{array}$$

TABLE 5. Challenge-Response Protocol

At the first step, A sends to B a fresh random number N_a (called a nonce and used as a challenge). At the second step, B returns the nonce signed using the appropriate private key k_b^{-1}. A checks if the message was signed using the private key associated with k_b.

DIDs and public keys are randomly generated by the document's owner in which the DID appears. DIDs and public keys are not necessarily correlated to the name of their owners, allowing them to receive anonymous services.

B. EHRS MANAGEMENT BASED ON INDY

Indy can provide an interesting EHRs management, where the patients hold wallets containing their attributes.

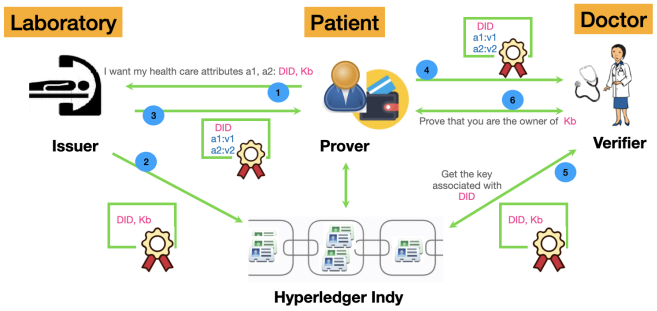

FIGURE 2. EHRs Management Based on Indy

Assume that a patient Bob, visited a laboratory L, and later he wants to give the doctor Alice access to some of his attributes produced by L. The main steps of the interaction between Bob, L, and Alice are shown in Figure 2.

1) After visiting any EHRs attribute provider, such as a laboratory, the patient can ask any time, even remotely, for his attributes. He sends a request containing a DID and a public key kb. The laboratory verifies the identity of the patient and asks him to prove that he is the owner of the public key kb using the challenge-response protocol shown in Table 5.
2) Once the patient's identity and ownership of the public key Kb have been proven, the laboratory creates a certificate attesting that DID belongs to the owner of kb and saves it in the hyperledger Indy as shown in Table 6.
3) The laboratory creates a signed document containing the DID, the requested attributes, their values and sends it to the patient.
4) The patient saves these attributes in his wallet as shown in Table 6 and can provide them for any future need. For example, Suppose that a doctor requests these attributes, then the patient uses his wallet and provides them.
5) The doctor Alice gets the public key Kb associated with DID from Indy.
6) Alice asks the patient Bob to prove that he is the owner of the kb by using a challenge-response protocol shown in Table 5.

This simple architecture provides numerous advantages, as shown in Table 7. Finally, it is worth mentioning that Indy also handles the revocation of DIDs in case a wallet is lost or hacked.

C. MODULE I: ACCESS CONTROL DELEGATION

We want to improve the previous version of EHRs access control management as follows:

Data \ Location	Patient Wallet	Indy	IPFS
EHRs	✓		✓
DID-Key certification		✓	

- ✓means that the data given in line is saved in the location given in the column.
- An empty case means that the data given in line is not in the location given in the column.

TABLE 6. Data Location

	Patient Controls Access to his EHRs	Anonymous Healthcare Service	EHR-Availability	EHR-Confidentiality	EHR-Integrity	EHR-Access Control Delegation	EHR-Zero-Knowledge Proof	EHR-Emergency Access	Scalability	Interoperability	Remote Health Service
Indy EHRs	✓	✓	+-	+-	+-		✓	✓	✓	✓	✓

- ✓means that the approach given in the line has the feature given in the corresponding column.
- +- means that this feature is not always present, it depends on the provider-side security level.
- An empty case means that the proposed approach has not the feature given in the corresponding column.
- ✓: The red color for the check mark indicates that this feature is provides by this approach but it is not the case for those presented in state of the art (Table 3).

TABLE 7. Features of the Solution Based on Indy

- The patient does not necessarily need to carry his attributes in a wallet. He can get them any time by contacting their providers.
- The patient can delegate his access control rights to any trusted third party as shown in table 8. A delegation is a kind of certificate that associates a DID with a set of attributes and a set of public keys. More precisely, a delegation has the form shown by Equation (1) and states that the attributes $at_1, \ldots, at_n$ of the patient known by DID can be accessed by anyone having a key in $pk_1, \ldots, pk_m$ until a deadline $Time$.

$$\underbrace{DID, at_1, \ldots, at_n, pk_1, \ldots, pk_m, Time}_{D}, \{H(D)\}_{k_b^{-1}} \quad (1)$$

The hole delegation policy is specified by a list of rules given in the format of Equation (1) which corresponds to the ABAC model [36]. In fact, each rule specifies the subjects by their public keys ($pk_1, \ldots, pk_m$), the objects by their names ($at_1, \ldots, at_n$), the action (read) is implicit, and the context by ($Time$). Of course, we can enrich the rule to take into account other contexts attributes (location, etc.). Also, instead of fixing the public keys of the subjects, we can fix some of their attributes (role, department, hospital). In this case, the subjects need to prove their attributes before getting access to the specified objects. The subject attributes and the proof of their ownership could be handled using Indy in the same manner as proving the ownership of others medical attributes.

Action on EHRs \ Roles	Patient	EHR provider	EHR consumer
Creation		✓	
Encryption		✓	
Upload		✓	
Access	✓	✓	after patient's permission
Delegate permission	✓		
Modify		✓	

- ✓means that the role given in the column is allowed to do the action given in the line.
- An empty case means that the action is not allowed for the role.

TABLE 8. Roles and Actions

The architecture of the updated solution is as shown in Figure 3, and it is used as follows:

1) The patient asks the laboratory to attest that he is the owner of the pair (DID, kb). Also, he provides the identity that allows the laboratory to identify him; this could be another DID.
2) The laboratory asks the patient to prove that he is B and he is the owner of kb. B can use any DID related to his public key, which the laboratory can know him.
3) The laboratory certifies the link between DID and kb and saves it in Indy as shown in Table 6. At the same time, it certifies and keeps secret the link between DID and B (i.e., $\{B, DID\}_{k_l}$). This link can be saved locally by the laboratory or encrypted using its public key and returned to the patient. Later, when the doctor asks for attributes related to DID, the link between DID and B (i.e., $\{B, DID\}_{k_l}$) should be provided or recovered.
4) The patient can add delegation related to any subset of his attributes using the Hyperldeger Fabric.
5) The patient visits the doctor and provides him the DID and the list of attributes $at_1, \ldots, at_n$ that he can obtain from the laboratory.
6) The doctor gets the public key kb associated with DID from Indy as well as the service endpoint of the trusted provider that certifies this link.
7) Using the public key kb, the doctor asks the patient to prove that he is its owner using the challenge-response protocol given by Table 5.
8) Using the service endpoint of the laboratory, the doctor sends a request to get attributes $at_1, \ldots, at_n$ associated with DID and provides his public key ka.
9) The laboratory gets the delegation from the hyperledger Fabric.
10) The laboratory gets the public key kb associated with DID from Indy and checks that the delegation obtained in the previous step is valid (signed using the private key associated with kb) and appropriate (it contains the key ka).
11) The laboratory gets the identity B of the patient associated with DID, searches for his attributes, and

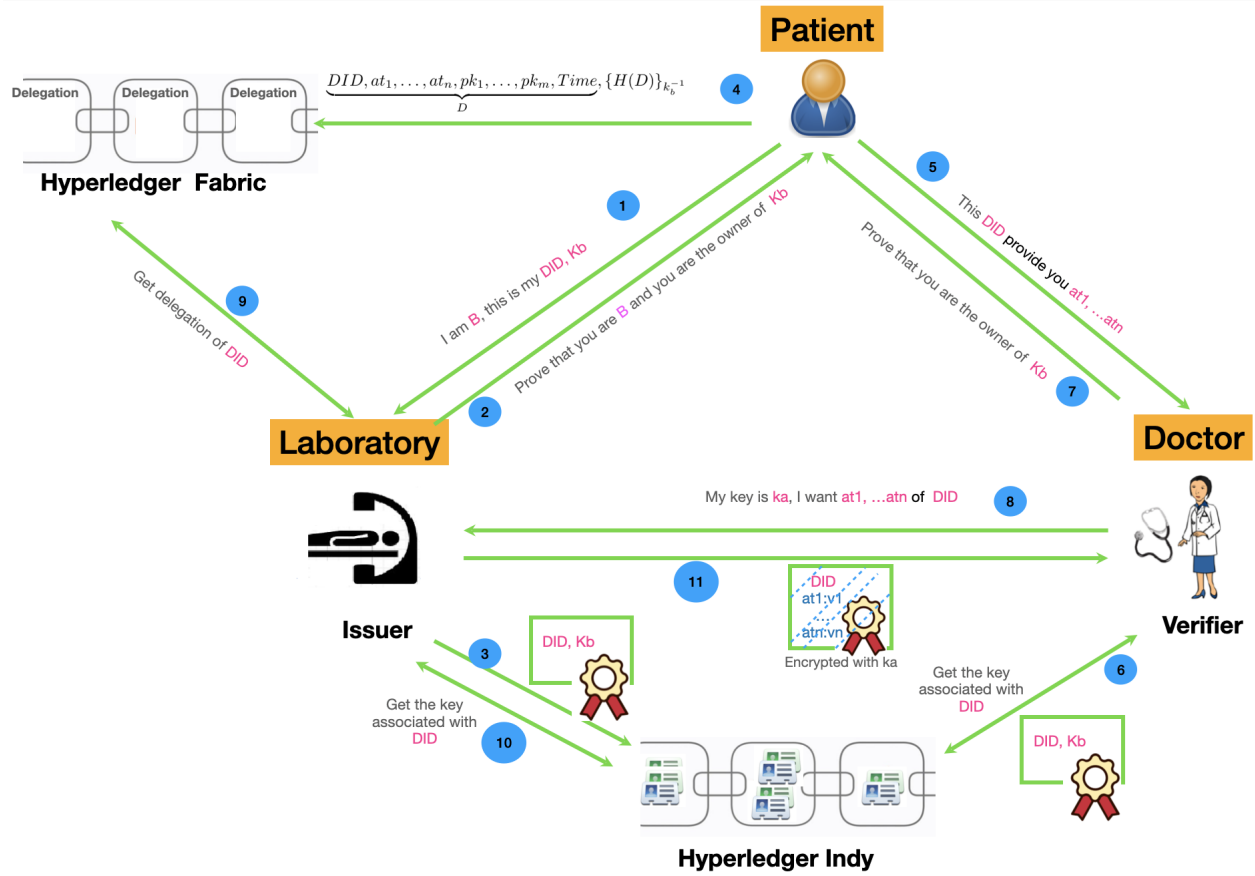

FIGURE 3. EHR Access Control Delegation Based on Indy and Fabric

provides them to the doctor encrypted by his public key ka as shown in Table 9. Please notice that, for efficiency reason, public keys are not usually used to encrypt data except keys and hashes. So, encrypting EHR record R with a public key k_a implicitly means that $\{R\}_k, \{k\}_{k_a}$ where k is a random private key.

	During transmission (Provider->Patient)	Wallet	During transmission (Patient->Consumer)
EHR protection	public key of Patient	public key of Patient	public key of Consumer

TABLE 9. EHR Protection

The updated solution has the features given in Table 10.

D. MODULE I: INCREASE INTEGRITY, AVAILABILITY AND CONFIDENTIALITY BY USING IPFS AND FABRIC

In the previous solution, the EHRs records were stored in the health care provider's local database. This means that the integrity, confidentiality, and availability of these EHRs depend on the provider side's security level who is continuously under cyberattacks. To improve the protection of EHRs, health care providers can save them in a distributed database such as the Interplanetary File System (IPFS) as shown in Table

	Patient Controls Access to his EHR	Anonymous Healthcare Service	EHR-Availability	EHR-Confidentiality	EHR-Integrity	EHR-Access Control Delegation	EHR-Zero-Knowledge Proof	EHR-Emergency Access	Scalability	Interoperability	Remote Health Service
Indy+Fabric EHRs	✓	✓	+-	+-	+-	✓	✓	✓	✓	✓	✓

- ✓means that the approach given in the line has the feature given in the corresponding column.
- +- means that this feature is not always present, it depends on the provider-side security level.
- An empty case means that the proposed approach has not the feature given in the corresponding column.
- ✓: The red color for the check mark indicates that this feature is provides by this approach but it is not the case for those presented in state of the art (Table 3).

TABLE 10. Feature of the Architecture Based of Indy and Fabric.

6. EHRs provided by a laboratory having a public key k_l is saved in IPFS as a file F containing $\{EHR\}_k, \{k\}_{k_l}$ where k is a random symmetric key. If we also want to give access to users having public keys $pk_1, \ldots, pk_n$, then we save a file F

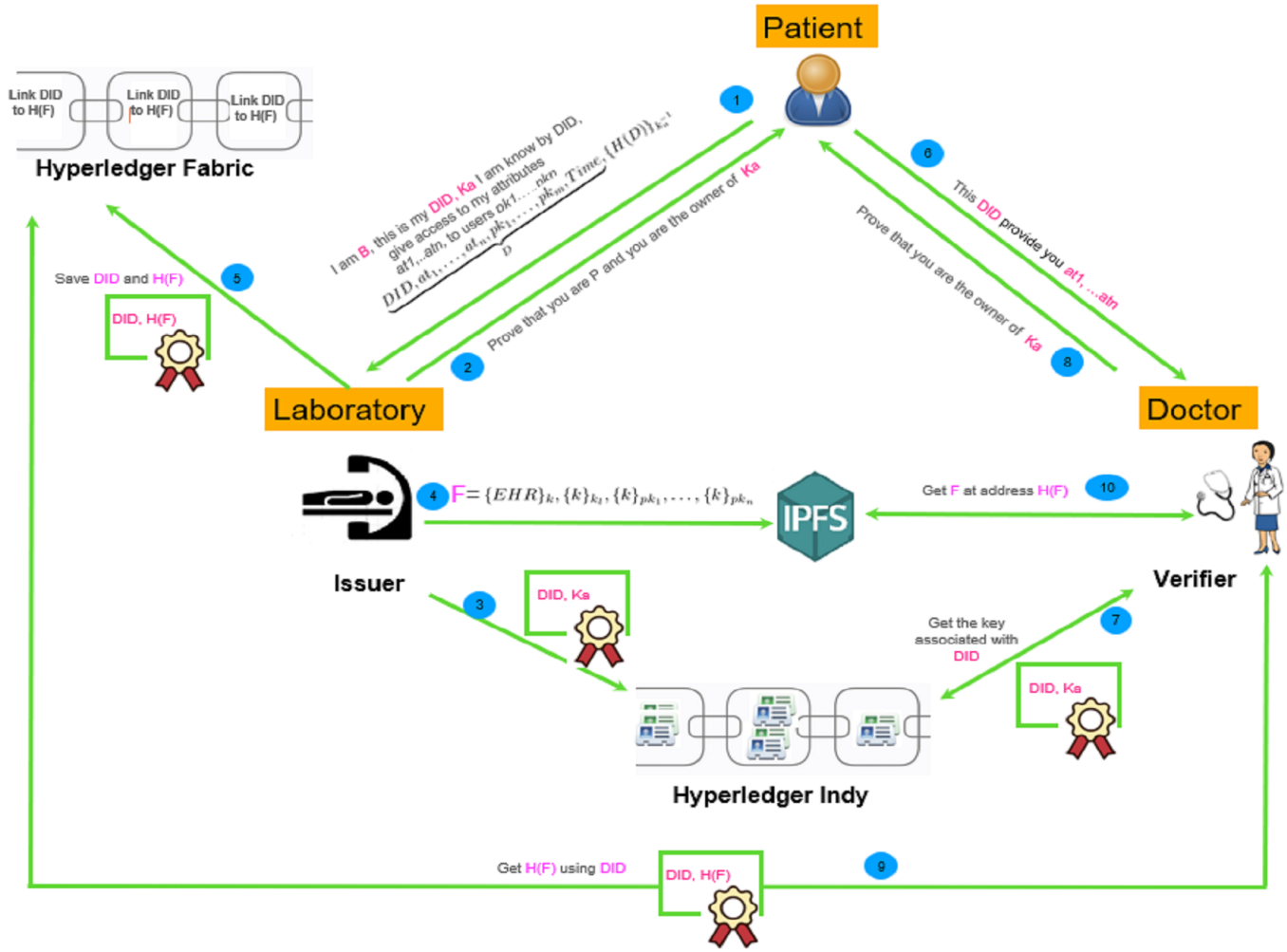

FIGURE 4. Using IPFS to enhance the security of EHRs

containing $\{EHR\}_k, \{k\}_{k_l}, \{k\}_{pk_1}, \ldots, \{k\}_{pk_n}$. This transparent protection technique gives us more confidence that the EHR confidentiality is appropriately protected. Also, IPFS provides a good availability since it is a distributed database with sufficient redundancy allowing access to files even if some storage nodes become offline. Moreover, to avoid the doctor directly contacting the laboratory to get the address of the file F containing the EHRs of a patient having identified by DID, we save $(DID, H(F))$ in the hyperledger Fabric. Using Fabric for this purpose, we increase the integrity of F and its availability. The new architecture is as shown in Figure 4. More precisely, the new solution works as follows.

1) First, the patient visits a laboratory and gets some service that creates new health care attributes, $at_1, \ldots, \ldots, at_n$. At any moment later, he can send a request to associate a DID to any parts of his attributes. He can also, at the same time or any moment later, send a delegation $\{EHR\}_k, \{k\}_{k_l}, \{k\}_{pk_1}, \ldots, \{k\}_{pk_n}$ to the laboratory. If the patient wants to give new principal access in the future, he can send a new delegation to the laboratory, and a new entry will be added in the IPFS, or we update the previous one.
   Notice also that we can handle the delegation as in the previous architecture, but with IPFS, we can immediately concretize it by giving access to the keys protecting the EHRs to all the authorized principals.
2) The laboratory checks the patient's identity and verifies that he owns the public key kb associated with DID.
3) The laboratory adds to Indy a certificate linking DID to kb.
4) The laboratory creates the file F containing the requested attributes encrypted with its key and all the key given within the delegation and saves it in the IPFS.
5) The laboratory adds to Fabric a certificate linking DID to $H(F)$ ($H(F)$ is the address of F in the IPFS).
6) The patient contacts the doctor Alice and asks for a health care service and provides the DID associated

with his attributes that he wants to share with her.
7) Alice gets the public key kb associated with DID from the hyperledger Indy.
8) The doctor asks the patient to prove that he owns the public key kb.
9) The doctor gets $H(F)$ associated with DID from the hyperledger Fabric.
10) The doctor gets the file F having the address $H(F)$ from IPFS.
11) The doctor uses her public key to decrypt the EHRs of the patient.

The updated solution has the features given by Table 11:

	Patient Controls Access to his EHR	Anonymous Healthcare Service	EHR-Availability	EHR-Confidentiality	EHR-Integrity	EHR-Access Control Delegation	EHR-Zero-Knowledge Proof	EHR-Emergency Access	Scalability	Interoperability	Remote Health Service
Indy+Fabric+IPFS EHRs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

- ✓means that the approach given in the line has the feature given in the corresponding column.
- +- means that this feature is not always present, it depends on the provider-side security level.
- An empty case means that the proposed approach has not the feature given in the corresponding column.
- ✓: The red color for the check mark indicates that this feature is provides by this approach but it is not the case for those presented in state of the art (Table 3).

TABLE 11. Feature of the Architecture Based of Indy, Fabric and IPFS.

Please notice that neither the blockchain nor the IPFS allow the modification of EHRs. This useful property allows to keep track of the healthcare history of the patient. If the value of any health attribute is changed, a new record is added to IPFS.

The difference features supported by the proposed architectures are summarized in Table 12:

Features / Architecture	EHR-Availability	EHR-Confidentiality	EHR-Integrity	Delegation	EHRs-Location	Delegation Location
Indy	+-	+-	+-		Patient's wallet	
Indy + Fabric	+-	+-	+-	✓	Patient's wallet	Fabric
Indy + Fabric + IPFS	✓	✓	✓	✓	IPFS	Fabric

- ✓means that the approach given in the line has the feature given in the corresponding column.
- +- means that this feature is not always present, it depends on the provider-side security level.
- An empty case means that the proposed approach has not the feature given in the corresponding column.

TABLE 12. The Differences Between the Proposed Architectures

Table 13 resumes the main computations (Hash, Signature, Signature Verification, Encryption, Pair Key Generation, Random Number Generation) performed by the different actors for the three proposed architectures. It shows three values (blue, red and green) corresponding to the three proposed architectures (Indy, Indy+Fabric, Indy+Fabric+IPFS). When the values are absent, it is implicitly means that we have 0,0,0.

E. HANDLING EMERGENCY ACCESS

Different techniques can be used to handle emergencies:

- The Decentralized Identifier (DID) used in Indy is a W3C standard 39. The main information in a DID record is the DID number and its owner's public key. However, it may contain many other information useful for different contexts. For example, we find the authentication protocols that the owner can use with verifiers. Also, the DID record may contain delegation information giving other public keys of other subjects that can act on behalf of the owner and can be used for emergency access. In addition, the DID record has been designed to be extensible, so that we can add other fields useful for our specific contexts. In particular, we can add fields the contact information related to delegated persons for emergency that we have given their public keys in the delegation filed. Another interesting feature provided by the DID is the ability to specify different verification and authentication methods. We can for example, use a threshold signature. In this case, we can specify n keys in the delegation part, and require that at least k of them need to sign a challenge to able to behave on behalf of the owner. Using this mechanism, it is even possible to revoke a key according to the owner requirement.
- We can simply use the secret splinting technique to achieve this goal. The idea is as follows: To be able to act on behalf of Bob during emergency, trustees need to know the DID of the vital attributes and the private key k_b^{-1} associated to it. However, Bob does not want to give k_b^{-1} to a particular trustee, but he allows any subgroup of trustees, with a predefined size, to reveal this private key. To this end, we can use the threshold cryptography (also called secret splitting) like [40]: the key k_b^{-1} is encrypted by a random secret key k that can be revealed only by a group of trustees having the appropriate size. A common way to achieve this goal is to generate a random polynomial $P(x)$ of degree $l-1$, where $k = P(0)$, then we generate n points in the polynomial. Every trustee receives only one point on the polynomial. In this case, we are sure that at least l users are needed to rebuild the polynomial and find the value of $k = P(0)$ using the famous Lagrange equations [41]. Of course, hospitals should know the contact of the trustees of Bob. For this reason, Bob needs to register them somewhere (e.g., hospital information systems) so that they will be found and contacted quickly. It will also

	Step / Computation	1	2	3	4	5	6	7	8	9	10	11
Patient	Hash	0,0,1		1,1,1	0,1,0	0,1,0	1,0,0		0,1,0	0,0,1		
	Signature or Private Key Encryption	0,0,1	0,1,1		0,1,0		1,1,0		0,0,1			
	Signature Verification or Public key Encryption			1,0,0								
	Symmetric Key Encryption/ Decryption											
	Key Generation	1,1,0										
	Random Number Generation	1,1,0										
Laboratory	Hash	1,0,1	1,1,1	1,1,1		0,0,1						
	Signature or Private Key Encryption		1,0,0	0,1,1		0,0,1						0,1,0
	Signature Verification or Public key Encryption	1,0,1	0,0,1		0,0,n					0,1,0		
	Symmetric Key Encryption/ Decryption				0,0,1							
	Key Generation											
	Random Number Generation		1,1,0									
Doctor	Hash				1,0,0	1,0,0	1,1,0	0,1,1	0,0,1	0,0,1		0,1,1
	Signature or Private Key Encryption										0,0,1	
	Signature Verification or Public key Encryption				1,0,0	1,0,0	1,1,0	0,1,1	0,0,1	0,0,1		0,1,0
	Symmetric Key Encryption/ Decryption										0,0,1	
	Key Generation											
	Random Number Generation						1,0,0	0,1,0	0,0,1			

TABLE 13. Computations Performed in the Three Proposed Architectures (Indy, Indy + Fabric and Indy + Fabric +IPFS)

be a good idea that hospitals will be one of the trustees. Obviously, the emergency procedure (finding trustees, contacting them, recovering the key, and revealing vital attributes) should be automated to do not waste any precious second.

F. MODULE II: WEBAUTHN FOR MORE SECURITY

In the previous architectures, the patient needs to protect his private keys appropriately in his computer or smartphone. However, this computer or smartphone can be infected by spyware that can steal these keys. To eliminate this risk, we can adapt Webauthn [31] or Fido U2F [28] to use an external USB security token that holds all the patient's private keys and does all the required cryptographic operations.

G. PERFORMANCE AND COST (GAS)

Fabric uses the KAFKA-consensus algorithm [32] whereas Indy uses the Redundant Byzantine Fault Tolerance (RBFT) [33]. As stated in [34], both of them are permissioned-voting based and provide a good speed and finality (approved blocks could not be revoked once committed). For instance, according to the official website of Fabric [5], the supported transaction speed is 3 000 transactions per second. It is a fast blockchain compared to Bitcoin (maximum of 7 transactions per second) or Etherium (45 transactions per second). For a comparison, according to [35], Visa handles 1700 transactions per second. Regarding Indy, the supported transaction speed, according to [37], is 100 transactions per second.

Both of Fabric and Indy do not require a gas system like Etherium neither a mining process like Bitcoin. Their consensus algorithms are executed by only some specific trusted nodes.

V. PROTOTYPE : A PROOF OF CONCEPTS

As a proof of concept, we implemented a prototype of our health framework using Ubuntu 18.04.6 LTS (Bionic Beaver), Hyper-ledger Fabric 2.3.1, and Hyper-ledger Indy. Some screenshots are given by Figures 5, 6, 7, 8, 9, 10 and 11 to better clarify the idea .

A. EHR BASED ON THE HYPER-LEDGER INDY

We started by implementing the solution that manages EHRs based on the Hyper-ledger Indy summarized in Figure 2 and detailed in section IV-B. Here are the various interfaces of the prototype involved during the different steps of the approach.

Suppose that the patient Bob visits a laboratory L for the first time to receive some medical analysis. Usually, after his visit, a new record is created containing an association between the Bob public key and his set of medical attributes. Bob is not requested to provide his name; he can stay anonymous by only proving that he owns the private key associated with his provided public one. This verification is done using the protocol of Table 5. At the end of this visit, the local database of laboratory L is updated by the following entry, where v_i is the value of the attribute a_i. The public key kb_0 will be an identifier of the patient Bob in laboratory L. It is not necessarily unique and can change from one visit to another.

PUBLIC-KEY	Date	Attribute a_1	...	Attribute a_n
⋮	⋮	⋮	⋮	⋮
kb_0	$2021-12-01$	v_1	...	v_n

Similarly, Bob saves an association between his keys and the laboratory name in his secure local wallet. His wallet will be updated using the following entry:

Healthcare-Provider	PUBLIC-KEY	PRIVATE-KEY	Date	Attributes
⋮	⋮	⋮	⋮	⋮
L	kb_0	kb_0^{-1}	$2021-12-01$	$a_1,\ldots,a_n$

Now suppose that Bob visits the doctor Alice, and he wants to show her some of his attributes provided by the laboratory L.

1) Bob sends a request to the laboratory to ask for a subset of attributes in $a_1,\ldots,a_n$ that are associated with kb_0. This request should contain the name of attributes, his owner (kb_0), a fresh public key kb, and fresh random DID as shown in Figure 5 . First, The laboratory checks the identity of Bob by asking him to prove that he owns

the public key kb_0 using the protocol of Table 5. Then, the laboratory asks Bob to prove that he is also the owner of the public key kb using the protocol of Table 5.

2) The laboratory creates a certificate attesting that DID belongs to the owner of kb and saves it in the hyper-ledger Indy as shown in Figure 6.
3) Then, the laboratory creates a signed document (patient credentials) for Bob containing the DID and the requested attributes with their values. After that, the laboratory sends these credentials to Bob, who saves them in his wallet as shown in Figure 7.
4) The patient Bob forwards the credential containing the attributes and their value to Alice.
5) Then, the doctor Alice accepts Bob's invitation as shown in Figure 8 and gets the public key Kb associated with DID from the hyper-ledger Indy.
6) Alice requests Bob to prove that he is the owner of kb as shown in Figure 9. This proof is made using the challenge-response protocol shown in Table 5 .

B. EHR BASED ON INDY AND FABRIC

Hereafter, we implement the solution of section IV-C , Figure 3 that Delegate Access Control on EHRs Based on Fabric. As in the previous case, during the first visit, Bob registers himself in the laboratory as the owner of some public key kb. He also creates a connection between the laboratory name and this public key in his local wallet.

1) Bob contacts the laboratory as the patient known by kb, asks this laboratory to attest that he is the owner of the pair (DID, kb) as shown in Figures 6. The laboratory asks Bob to prove that he is the owner of kb using the challenge-response protocol shown by Table 5 .
2) The laboratory certifies the link between DID and kb and saves it in the Hyper-ledger Indy as shown in Figure 6.
3) Bob can add delegation related to his attributes provided by the laboratory to the Hyperldeger Fabric as shown in Figure 10.
4) Bob visits the doctor Alice and provides her the DID and the list of attributes $at_1, \ldots, at_n$ that she can obtain from the laboratory as shown in Figure 11.
5) Alice gets the public key kb associated with DID from Indy as well as the service endpoint of the trusted provider that certifies this link.
6) Using the public key kb, Alice asks Bob to prove that he is its owner using the challenge-response protocol given by Table 5.
7) Using the service endpoint of the laboratory, Bob sends a request to get the attributes $at_1, \ldots, at_n$ associated with DID and provides his public key kb.
8) The laboratory gets the delegation from the hyper-ledger Fabric as shown in Figure 10.
9) The laboratory gets the public key kb associated with DID from Indy as shown in Figure 6 and checks that the delegation obtained in the previous step is valid (it is signed using the private key associated with kb) and appropriate (it contains the key kb).
10) The laboratory gets the identity kb0 of the patient associated with DID, searches for his attributes, and provides them to the doctor encrypted by his public key kb.

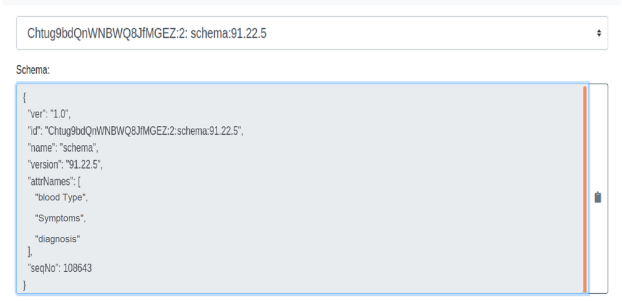

FIGURE 5. Bob Request to the laboratory for subset of her Attributes in Hyper-ledger Indy

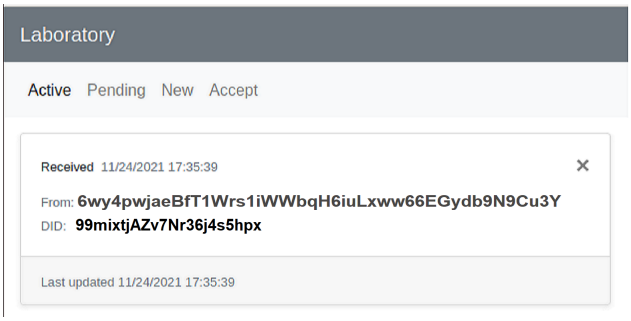

FIGURE 6. The laboratory creates a certificate attesting that DID belong to the owner of the public key

FIGURE 7. Bob's Credential in the Hyper-ledger Indy

VI. DISCUSSION

The main contribution of this paper is a modular architecture for EHRS. The core of this architecture is the Hyperledger Indy which is specially designed for the Self-Sovereign Identity (SSI) where individuals have full control of their digital identities (a set of attributes that includes medical ones.). Indy intrinsically provides many interesting features for the EHR management such as:

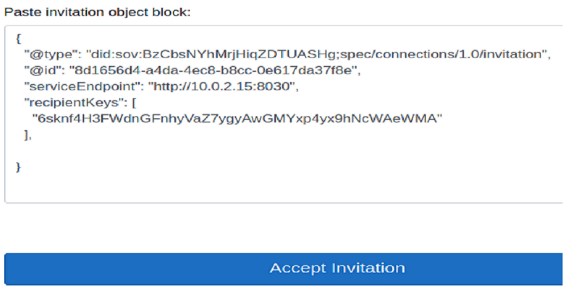

FIGURE 8. Alice accepts the Bob Invitation

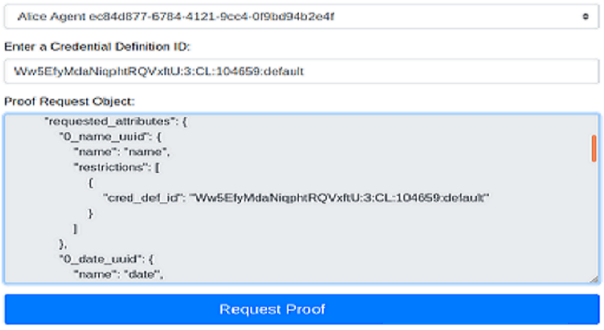

FIGURE 9. The doctor Alice asks Bob to prove that he is the owner of public key

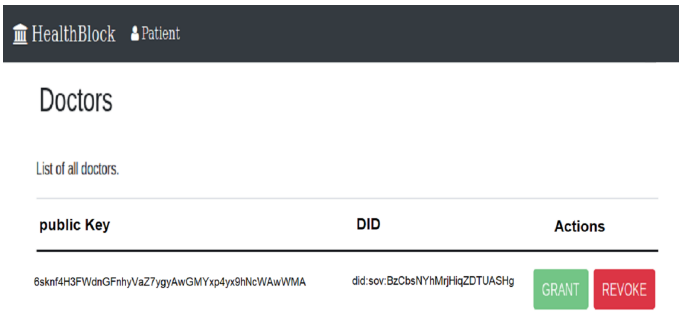

FIGURE 10. Bob delegates access related to his attributes

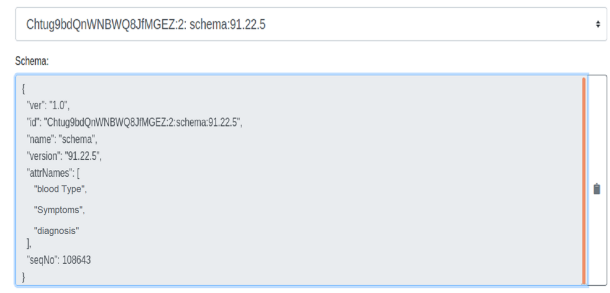

FIGURE 11. Alice provides Bob the DID and list of his attributes

- no one can access to healthcare attributes without the consent of their owners.
- patients can remotely prove that they are the owners of their EHRs.
- patients can remotely prove, using Zero-Knowledge Proof (ZKP), that a part of their healthcare attributes respect some properties without revealing their values (e.g., blood pressure is less than 130 mm Hg).
- patients can hide some attributes in an EHR while revealing others.

In addition, the initial architecture has been upgraded with Fabric to allow access control delegation. This second hyperledger is used for the storage of the patient's access control policies specified in the ABAC style. The delegation usually simplify the access control by reducing the intervention of patients. Using Fabric as a generic hyperledger, provides the availability, the integrity and the traceability of access control policies.

Finally, we upgraded the architecture by adding IPFS that can play the role of wallets for the patients and increases the availability, the integrity and the traceability of the patients' EHRs.

Compared to the state of the art, this new architecture provides more useful feature as shown by Tables 11 and 3.

VII. CONCLUSION

This paper proposes a modular framework called *HealthBlock* for collaborative sharing of EHRs based on Blockchain. The solution is mainly based on the hyperledger Indy providing a self-sovereign identity (SSI) to allow patients to have full control of their EHRs, scalability, interoperability, anonymous healthcare service. Also, we used hyperledger Fabric for access control management and Zero-Knowledge Proof of EHRs and used IPFS to increase Availability, Confidentiality, and Integrity. We can adapt Webauthn or Fido U2F to use an external USB security token that holds all the patient's private keys and does all the required cryptographic operations to allow the patient to protect his private keys in his computer or his smartphone.

REFERENCES

[1] IPFS. IPFS powers the Distributed Web.https://ipfs.io. Accessed March 7th, 2021.
[2] IPFS. Bitswap. Web https://docs.ipfs.io/concepts/bitswap/. Accessed May 11th, 2022.
[3] GIT. Distributed Is the New Centralized. Web https://git-scm.com. Accessed May 11th, 2022.
[4] D. Maziéres, M. Kaminsky, M. F. Kaashoek, E. Witchel "Separating key management from file system security". Proceedings of 17th ACM Symposium on Operating System Principles (SOSP '99), Kiawah Island, South Carolina, December 1999.
[5] Linux Foundation. Hyperledgre Fabric. https://www.hyperledger.org/use/fabric. Accessed March 7th, 2021.
[6] W3C.Verifiable Credentials Working Group. https://www.w3.org/2017/vc/WG/. Accessed January 2th, 2021.
[7] J. Sun, X. Yao, S. Wang, and Y. J. I. A. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in IPFS," vol. 8, pp. 59389-59401, 2020.
[8] Kumar, R., Marchang, N., Tripathi, R.: Distributed off-chain storage of patient diagnostic reports in healthcare system using IPFS and blockchain. In: COMSNETS, pp. 1–5. IEEE (2020).
[9] A. J. E. Khatoon, "A blockchain-based smart contract system for healthcare management," vol. 9, no. 1, p. 94, 2020.

[10] B. Shen, J. Guo, and Y. J. A. s. Yang, "MedChain: Efficient healthcare data sharing via blockchain," vol. 9, no. 6, p. 1207, 2019.
[11] S. Niu, L. Chen, J. Wang, and F. J. I. A. Yu, "Electronic health record sharing scheme with searchable attribute-based encryption on blockchain," vol. 8, pp. 7195-7204, 2019.
[12] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. J. I. a. Seneviratne, "Blockchain for Secure EHRs Sharing of Mobile Cloud Based e-Health Systems," vol. 7, pp. 66792-66806, 2019.
[13] A. Shahnaz, U. Qamar, and A. J. I. A. Khalid, "Using blockchain for electronic health records," vol. 7, pp. 147782-147795, 2019.
[14] M. G. Kim, A. R. Lee, H. J. Kwon, J. W. Kim, and I. K. Kim, "Sharing medical questionnaries based on blockchain," in 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2018, pp. 2767-2769: IEEE.
[15] W. J. Gordon, C. J. C. Catalini, and s. b. journal, "Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability," vol. 16, pp. 224-230, 2018.
[16] G. G. Dagher, J. Mohler, M. Milojkovic, P. B. J. S. c. Marella, and society, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," vol. 39, pp. 283-297, 2018
[17] K. Fan, S. Wang, Y. Ren, H. Li, and Y. J. J. o. m. s. Yang, "Medblock: Efficient and Secure Medical Data Sharing via Blockchain," vol. 42, no. 8, p. 136, 2018.
[18] J. Liu, X. Li, L. Ye, H. Zhang, X. Du, and M. Guizani, "BPDS: A blockchain based privacy-preserving data sharing for electronic medical records," in 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1-6: IEEE.
[19] A. Al Omar, M. S. Rahman, A. Basu, and S. Kiyomoto, "Medibchain: A blockchain based privacy preserving platform for healthcare data," in International conference on security, privacy and anonymity in computation, communication and storage, 2017, pp. 534-543: Springer.
[20] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and Trustable Electronic Medical Records Sharing using Blockchain," in AMIA Annual Symposium Proceedings, 2017, vol. 2017, p. 650: American Medical Informatics Association.
[21] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating Blockchain for Data Sharing and Collaboration in Mobile Healthcare Applications," in 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017, pp. 1-5: IEEE.
[22] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. J. I. A. Guizani, "MeDShare: Trust-less Medical Data Sharing Among Cloud Service Providers via Blockchain," vol. 5, pp. 14757-14767, 2017.
[23] T.-T. Kuo, H.-E. Kim, and L. J. J. o. t. A. M. I. A. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," vol. 24, no. 6, pp. 1211-1220, 2017.
[24] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using Blockchain for Medical Data Access and Permission Management," in 2016 2nd International Conference on Open and Big Data (OBD), 2016, pp. 25-30: IEEE.
[25] X. Yue, H. Wang, D. Jin, M. Li, and W. J. J. o. m. s. Jiang, "Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control," vol. 40, no. 10, p. 218, 2016.
[26] M. T. SandIkkaya, B. De Decker, V. Naessens, M. Szomszor and P. Kostkova. Privacy in commercial medical storage systems. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering; 2011; Vol. 69 LNICST; pp. 247 - 258.
[27] M. Masi and R, Maurer. On the Usage of SAML Delegate Assertions in an Healthcare Scenario with Federated Communities. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering; 2011; Vol. 69 LNICST; pp. 212 - 218.
[28] FIDO Alliance. Simpler, Stronger Authentication. https://fidoalliance.org. Visited on 2th, 2021.
[29] Sovrin Foundation. Sovrin. URL: https://sovrin.org (visited on December 23, 2020).
[30] Sovrin Foundation. Use case spotlight: The Government of British Columbia uses the SovrinNetwork to take strides towards a fully digital economy. URL: https://sovrin.org/usecase-spotlight-the-government-of-british-columbia-uses-the-sovrin-networkto-take-strides-towards-a-fully-digital-economy/ (visited on December 23, 2020).
[31] Webauthn. Using WebAuthn. https://webauthn.io. Visited on January 2th, 2021.
[32] J. Kreps, N. Narkhede and J. Rao, "Kafka: A distributed messaging system for log processing", Proc. NetDB, pp. 1-7, 2011.
[33] P. L. Aublin, S. Ben Mokhtar and Quéma. "RBFT: Redundant Byzantine Fault Tolerance". IEEE 33rd International Conference on Distributed Computing Systems, 2013.
[34] The Linux Foundation. "Hyperledger Architecture, Volume 1: Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus". 2017. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf (visited on Mars 20th, 2022).
[35] K. Sedgwick. "No, Visa Doesn't Handle 24,000 TPS and Neither Does Your Pet Blockchain". URL: https://news.bitcoin.com/no-visa-doesnt-handle-24000-tps-and-neither-does-your-pet-blockchain/ (visited on Mars 20th, 2022).
[36] Vincent C. Hu D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone. "Guide to Attribute Based Access Control (ABAC) Definition and Considerations". NIST Special Publication 800-162, 2014. https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf.
[37] Edx. "Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries & Urs". Online Source, last visit (April 2022). https://learnthings.online/other/2020/03/05/introduction-to-hyperledger-sovereign-identity-blockchain-solutions-indy-aries-ursa.
[38] IBM. "Behind the Architecture of Hyperledger Fabric", 2018. Online Source, last visit (April 2022). https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/.
[39] World Wide Web Consortium (W3C). "Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations". W3C standard 2021, Online resource, last visit (April 2022. https://www.w3.org/TR/did-core/#authentication
[40] A. Shamir . "How to share a secret" (PDF). Communications of the ACM. 22 (11): 612–613, 1979;
[41] J. P. Berrut and L. N Trefethen. "Barycentric Lagrange Interpolation" (PDF). SIAM Review. 46 (3): 501–517, 2004.

LEINA NAZAR is currently pursuing a Ph.D. degree in computer science with the Sudan University of Science and Technology. she received a Master's Degree in computer science from Karary University, Sudan, in 2017. she received the B.Sc. degree in Software Engineering from El-Mashreq university, Sudan, in 2011 .she has Membership of the Blockchain Council since 2020. she has Blockchain Developer certification from the Simplilearn platform since 2020. Her research topics cover computer security and blockchain technology.

DR MOHAMED MEJRI received his PhD. on the specification and analysis of cryptographic protocols from Computer Science and Software Engineering Department of Laval University, Canada . He also received an engineering degree in computer science from the National School of computer Science (ENSI), Tunisia, and master degree in computer science for Laval University, Canada. Currently, he is a full professor in Computer Science and Software Engineering Department of Laval University, Canada. His research topics cover computer security, formal methods and software engineering.