Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 9763937, 11 pages
https://doi.org/10.1155/2018/9763937

Research Article

Blockchain Technology: Is It a Good Candidate for Securing IoT Sensitive Medical Data?

Nabil Rifi,[1,2] Nazim Agoulmine,[1] Nada Chendeb Taher,[2] and Elie Rachkidi[1]

[1]*COSMO, IBISC Laboratory, University of Evry, Paris Saclay University, France*
[2]*Lebanese University, Faculty of Engineering and Azm Center for Researches, Tripoli, Lebanon*

Correspondence should be addressed to Nazim Agoulmine; nazim.agoulmine@ufrst.univ-evry.fr

Received 15 August 2017; Revised 19 December 2017; Accepted 25 June 2018; Published 5 December 2018

Academic Editor: Fernando De la Prieta

In the past few years, the number of wireless devices connected to the Internet has increased to a number that could reach billions in the next few years. While cloud computing is being seen as the solution to process this data, security challenges could not be addressed solely with this technology. Security problems will continue to increase with such a model, especially for private and sensitive data such as personal data and medical data collected with more and more smarter connected devices constituting the so called Internet of Things. As a consequence, there is an urgent need for a fully decentralized peer-to-peer and secure technology solution to overcome these problems. The blockchain technology is a promising just-in-time solution that brings the required properties to the field. However, there are still challenges to address before using it in the context of IoT. This paper discusses these challenges and proposes a secure IoT architecture for medical data based on blockchain technology. The solution introduces a protocol for data access, smart contracts and a publisher-subscriber mechanism for notification. A simple analytical model is also presented to highlight the performance of the system. An implementation of the solution as a proof of concept is also presented.

1. Introduction

IoT is taking over the world; it is estimated that the number of devices connected to the Internet forming the Internet of Things will reach 50 billion by 2020 [1]. One critical application is the eHealth smart homes. In fact, this technology allows monitoring elderly or individuals with diseases and automatically sending the data to a remote server for processing by doctors. This data is recorded in the so-called EMR (Electronic Medical Record).

Secure access to this EMR is problematic considering privacy issues, transparency, etc. This is why in order to develop secure and reliable solutions for eHealth smart homes, it requires unprecedented coordination and collaboration between all pieces of the system. All devices must work together and be integrated with all other devices, and all devices must communicate and interact seamlessly with remote systems and infrastructures in a secure way. Such a solution is possible, however it can be expensive and time consuming. Thus, there is a need for new ideas and new technologies that will drive IoT security towards a more decentralized model.

Having this huge amount of data, being centralized and sometimes monitored by one single provider, may create many problems. The cloud as a computing/storing technology cannot only by itself protect the security and privacy of its users. Using a decentralized approach for IoT network security is eventually an interesting way to solve many of the challenges IoT technology is facing today. Adopting a peer-to-peer model to handle billions of transactions between the billions of interconnected devices will decrease dramatically the costs of installation and maintenance of data centers and servers. It will also allow the distribution of storage and processing power on different devices and components of the network increasing the reliability of the system; e.g., the failure of one node will not cause the entire network to halt or collapse.

However, in order to establish well-defined peer-to-peer communication protocols, a whole other set of challenges will need to be addressed, mainly security and privacy. Some level

of peers validation and consensus must be reached in order to prevent spoofing and theft, an important characteristic that any peer-to-peer distributed solution must have, especially large scale IoT networks. A successful approach to decentralized control of IoT must not only be a peer-to-peer approach, but also a trust environment in which no participant needs to be trusted, and no single point of trust failure exists. Blockchain technology offers these features and proved its efficiency in financial applications such as Bitcoin [2, 3], and it can be of great value and importance in this domain. In this paper we discuss first important background to understand blockchain technology value as well as related works. Then, we present the specific problem we want to address in the context of IoT and finally propose a solution architecture and model. The proposed solution is based on smart contracts [4] and Publisher-Subscriber mechanism. In the following, we present a mathematical model to evaluate the performance of the system and its implementation. Finally, we end up with a conclusion and some future works.

2. Related Works

2.1. Background

2.1.1. Blockchain. In 2008 Satoshi Nakamoto introduced Bitcoin [5], a fully digital and decentralized cryptocurrency. In order to solve the double spending problem in Bitcoin, blockchain technology was introduced. It is a peer-to-peer decentralized distributed ledger that is replicated on all nodes participating in the system. It is a complete transparent technology that can show all the transactions that have been made since its creation, without tampering or fraud [6]. Blockchain is a group of blocks that are connected, each block to the one before. The first block is called the "Genesis" block and it is hard coded into the software. Think of the blockchain as a log whose records are batched into timestamped blocks, each block being identified by its cryptographic hash. Each block references the hash of the block that came before it. This establishes a link between the blocks, thus creating a chain of blocks or blockchain. Any node with access to this ordered, back-linked list of blocks can read it and figure out what is the world state of the data that is being exchanged on the network.

To understand how blockchain works, we will first explain how a blockchain network works. This is a set of nodes that operate and interact on the same network using the replicated blockchain copy each node holds. One node might be used as an entry point for multiple users or devices, but let first assume that each node transacts on the network by itself only. First of all, the interaction between users nodes and the blockchain is done via a pair of private and public keys.

The transactions are signed using the private keys, and then addressed by other nodes using the corresponding public keys [7]. The use of asymmetric cryptography is very important in terms of providing integrity, nonrepudiation, and authentication to the blockchain network. After a transaction is signed, it is broadcasted by the node to all other nodes it is connected to. The nodes, or peers, need to ensure that the transaction is valid, thus they need to trigger a validation process called "mining". A transaction in the blockchain can be defined as any transfer of data that has a value (Bitcoin, IoT data, etc.). After a transaction is validated via the mining process, it is then added to the blockchain as a new block of transaction and data. Since the blockchain is transparent, these transactions are available for all nodes connected to the network. These blocks are organized in a linear sequence that grows overtime, where each block contains a timestamp and the hash of the previous block. This organization of the blocks constitutes the so called blockchain.

When presenting blockchain technology, there are two main platforms to mention: Bitcoin blockchain which is the famous cryptocurrency blockchain, and Ethereum blockchain [8]. Ethereum is another public blockchain that allows users to implement smart contracts, create private blockchain, and test transactions. In our proof of concept implementation, we have used the Ethereum blockchain.

2.1.2. Proof of Work. It is used in the blockchain in such a way that each node participates to solve difficult mathematical puzzles that their solution validates blocks. This puzzle is usually a function of the previous blocks hash, in order to maintain the chronological order and solving the double spending problem. Once this puzzle is solved, there is a "winner" that publishes the solution. Once the solution is published, it is easy for other nodes to verify the solution. However, solving the puzzle is expensive when it comes to computational power. Once a majority of the peers accept the solution, all the nodes in the network begin working on the next block; and a repeating process takes place. Proof of work is one approach to reach consensus. It is very efficient in terms of securing the transactions and making the blockchain immutable, because it requires a great amount of resources, from computational resources to electrical and CPU powers. However, this approach has its weaknesses since it is inefficient for typical devices that are not very powerful. The motivation behind proof of work is that the nodes which invest significant resources into the network are less likely to attack or cheat.

2.1.3. Proof of Stake. Proof of stake is another approach that aims to solve the computational power problem. In order to validate a certain block or transaction, the amount of cryptocurrency is what matters. 51 % of the digital currency owners need to agree on the current state. The reason behind proof of stake approach is quite simple. The higher is the stake in the system, the more expensive it is for nodes to maintain a secure network. Proof of stake is less secure than proof of work, however it provides a more efficient approach to the lack of computational power issue.

2.1.4. Smart Contracts. It is possible to include code to be executed in the blockchain allowing general purpose computation. Similar to a contract between any two individuals, the smart contract is a piece of code that can have conditions to be trigerred and actions to execute if conditions are verified. The importance of smart contracts comes with their capabilities to manage the interactions between nodes and participants of the system based on the data. Smart contracts like any

other node have addresses in the blockchain. Triggering a smart contract is done by addressing a transaction to it. The security of these contracts is however very important in the blockchain [9].

2.1.5. Mining. Mining is one of the most important processes in blockchain technology. Mining is the act of validating new blocks so that they could be added to the blockchain. Mining is done by providing "proof of work" to validate a block; i.e., each block contains a mathematical puzzle that needs to be solved by the miner in order to provide proof of work. This process can be very expensive regarding computational power. Miners are individuals or organizations having dedicated considerable computational power for mining and maintaining the blockchain.

2.2. Related Works in the Integration of IoT, Blockchain, and eHealth. After an exhaustive investigation of this field, we found out that blockchain is a great candidate for future decentralized IoT architectures and models. For example, authors in [1] have defined a complete IoT architecture composed of three layers, where blockchain is used as the storage layer. The authors also defined a data management and a data sharing protocol, along with a study on different mechanisms and their impacts on the system i.e. direct blockchain access, server client access, and publisher/subscriber access. Based on this study, we found that the publisher/subscriber mechanism is very important and the most flexible mechanism. Therefore, we decided to adopt it in our solution. Blockchain technology is also considered in several ehealthcare solution. As example, authors of paper [10] have highlighted the general role and future impact of blockchain technology in healthcare. In the paper [11], a smart contracts based solution has been defined by the authors to manage the access to electronic medical records. Contribution in [11] is also important since it has defined the basic use of smart contracts in managing relationships between different parties of a blockchain. Another study that focused on the use of smart contracts in the context of IoT [12] and presents important aspects and different points of view regarding the benefit of using smart contracts. A use case of Internet of Things in the context of eHealth is to have medical sensors composing a Wireless Body Area Network (WBAN) and generating data. This approach is discussed in [13]. Authors have used a cryptographic approach applied to the blockchain and an intermediate device to manage data flow and connection to the blockchain. Authors of papers [14, 15] have also described a cryptographic approach where blockchain is implemented as a solution to protect personal data. Authors in [16] has discussed the design issues of integrated IoT and blockchain. The discussion is very helpful to understand the different possible implementation options (fully centralized, pseudocentralized, distributed, and fully distributed). Paper [17] discusses the data privacy and integrity issues in the area of eHealth and more particularly clinical trials. Authors have proposed in this contribution a solution that integrates IoT and blockchain to achieve their objective. This work is highly related to our research but focuses more on the relation between parties in the context of clinical trials. The paper [18] is not fullyin the area of eHealth but it presents a comprehensive solution to secure data in the context of smart homes. Authors of the paper have proposed a solution based on the integration of IoT and blockchain to prevent security attacks against external attacks by monitoring the IoT devices transactions. Our research aims to take benefits from all these state of art contributions to derive a well-defined solution to use blockchain technology for IoT data access protection. In the next section, we will highlight the exact challenges to face when applying blockchain to IoT.

We would like also to mention that this paper is an extended version of our previous work presented in [19].

2.3. Challenges in Using Blockchain Technology with IoT. As previously mentioned, blockchain is a peer-to-peer decentralized technology that provides transparent and powerful certification mechanisms while removing the need to trust external third parties. While this is a very important advantage, it comes also with some important challenges. First of all, with the intrinsic architecture of the blockchain, the ledger is replicated on all nodes connected to the blockchain. Therefore, the storage of large amount of data is inefficient if it is in the blockchain itself. This is indeed a big problem for IoT devices such as typical sensors that do have limited computing and storage resources. In addition, since there is no third parties involved in the system, every block added to the blockchain needs to be validated by each of its nodes. This is called the "'Mining'" process as previousely explained and it requires high computational power demand [20] to runencryption algorithms. As the IoT ecosystem is very diverse and could consist of heterogeneous devices with very different computing capabilities, all these devices will not be able to execute the same encryption algorithms at the desired speed and this should be taken into account in the proposed architectural solution. Another characteristic of IoT that renders the problem even more complicated is the scalability problem. Indeed, when the number of IoT nodes increases, the number of generated data will also increase and therefore the the number of transactions to the blockchain will also increase. The mining and validation process of the blockchain will then take more time to complete and this should be mitigated as much as possible to acheive the scalability property of IoT systems. In this paper, we aim to address specifically this mining problem and the time taken by the system to complete the transactions. We have therefore focused our research on the factors and parameters that might have the higher impact on the complexity of the mining process such as the block size and the block time. In the next section, we will discuss the proposed architecture of the distributed IoT system followed by the proposed solution mechanisms.

3. Proposed Architecture

The main idea of our approach to solve the problem of the high computational power needed in order to communicate with the blockchain is to introduce a centralized/decentralized combined architecture, i.e., to introduce intermediate servers between IoT devices and the blockchain.

A very good candidate for such an intermediate system is a cloud edge server, easily accessible by clients, and with high computational power. The other proposition is to use of a publish-subscribe notification mechanism. In particular, we have taken into account the results obtained from the authors of paper [1] who have shown that the publish-subscribe mechanism is very efficient in terms of filtering data and introducing intermediates to address the high demand of electrical and computational powers from blockchain mechanisms. In addition, we propose to use in our solution smart contracts to maintain rules, authentication, and communication between the different nodes and parties of our system. Finally, since the transaction of messages with large amount of data highly negatively impact the performance of the blockchain, we propose to use an off-chain database to store the data, in our case we have used IPFS [21] as peer to peer storage system. The Figure 1 presents a high level diagram of the proposed architecture identifying the different actors, the blockchain and the IoT devices.

In the next sections, we will first present the considered scenario then the details of the proposed architecture in terms of components and their interactions.

4. Proposed Solution

4.1. Defining the Scenario. To design our solution, we have first specified a potential real life scenario to help identifying the required components and functionalities. This scenario is related to the eHealth smart home and the remote monitoring of patients. We considered a home equipped with numerous connected sensors (i.e., connected things) that collect environmental data from the home (e.g. temperature, humidity, etc.) as well as medical sensors that collect health data from inhabitant (e.g., heart beats, ECG, etc.).

The problem we aimed to address is the following: how can the monitored persons can ensure that the data that is collected from their private environment and their own body and sent outside their home is only accessed by authorized persons (e.g., authorized doctors) and that it is not altered for any malicious reason by a third party ?

The direction that has been followed to solve this problem is actually to protect the location where the collected data is stored. To acheive this objective, we will highlight how the components of our blockchain IoT architecture (blockchain, smart contracts, intermediate servers, and off-chain database) could be efficiently glued together for this purpose. The main actors of this scenario are the publishers of data and the subscribers to these data while the end-providers and consumers are the IoT devices and the end-users (data generators and data consumers). Figure 2 presents our solution to be deployed in the smart home that is derived from the more general architecture presented in Figure 1.

The main idea of the solution is highlighted in Figure 2 that is to overcome the limitation of connected objects in a smart home not able to directly connect to the blockchain due to their limited processing capacity and energy power. This is achieved using intermediary edge system that connects the smart home to the blockchain and called the gateway. This gateway is expected to be more powerful than normal sensors and will play the role of the publisher to the blockchain. Eventually, it will publish the data received from the set of connected sensors in the smart home to the blockchain. The owners of this data are the monitored persons (e.g. elderly) who live in their homes and have access to this data. These persons can specify a set of authorized individuals or organisation that are authorized to access this data.

4.2. Defining the Smart Contracts. In order to specify the formal relation between data owners, consumers, and clients, smart contracts are the corner stone of our blockchain IoT system.

We have defined 3 types of contracts. The proposed smart contracts are presented in Figure 3.

4.2.1. The Publisher Contract. The first contract is the publisher contract that is deployed by the publisher. One can think about contracts as functions that take input, check it, and sendback results. When an individual subscribes to the system and connects his smart home gateway to the blockchain, he must first specify his gateway or publisher contract. Once accepted, he receives an unique ID mapped to his blockchain address (Ethereum [22] address in the case of Ethereum blockchain). He must also specify the list of IoT devices to connect to the blockchain. This can be done by names to ease access and comprehension of the generated data. In addition, he must specify a type of sharing mechanism to manage the publisher/subscriber relationship, and finally, he mustspecify the list of addresses that have permission to access the data (usually these are the addresses of subscribers who are authorized to access the data).

4.2.2. The Subscriber Contract. The second contract is the subscriber contract. It must contain the address of the subscriber in the blockchain, and the list of publishers to which it subscribes. It must also specify the specific list of sensors to subscribe to. The sensors can be chosen by type, by name or using a wildcard to select all the available sensors connected to a particular publisher. This is the critical component of the publisher-subscriber algorithm, since based on the information stored in that contract, the generated data can be filtered before sending it to a subscriber.

4.2.3. The Client Contract. The third contract is the client contract. It serves as a mapping contract between normal nodes, or clients connected to the blockchain, and their respective subscriber contracts. It contains the client name, so that it would be simpler for clients to communicate between each other using a frontend application. This name is mapped to the corresponding address in the blockchain.

4.3. The Protocol Description. Before describing our protocol, we first define some notations that will be later used in the paper:

(i) P_k: publisher number k

(ii) d_i^k: device i connected to publisher k

(iii) S_j: subscriber number j

(iv) C_j: client number j

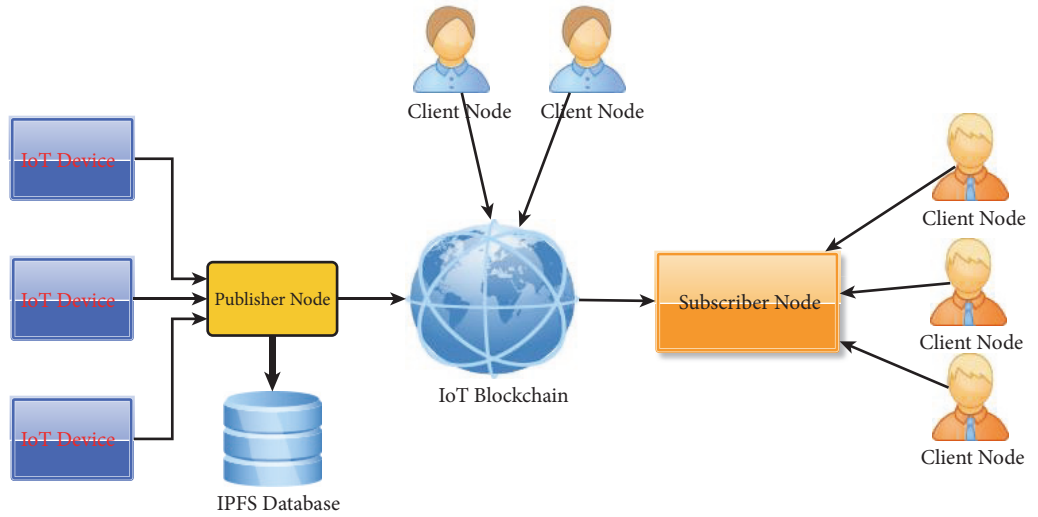


Figure 1: Proposed architecture presenting IoT devices connected to the Publisher node itself connected to the blockchain. On the other side, clients are either connected to subscriber nodes or directly to the blockchain.

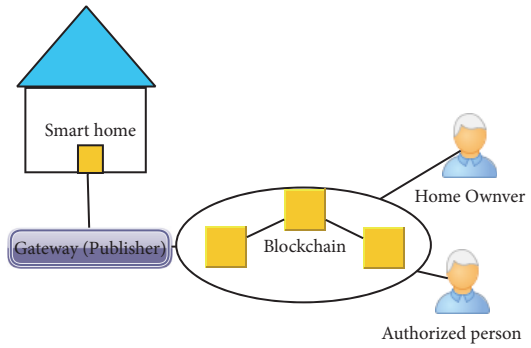


Figure 2: An architecture for the smart home blockchain IoT that identifies the gateway to the blockchain, the owners, and the authorized persons to access the private data.

```
1: function PUBLISHER                                              ▷
2:     initializePublisher
3:     while (eventfromanIoTdevice = True) do
4:         Da ⟵ ReadLocalData(d_i^k)
5:         a ⟵ SendingIoTDeviceID(Da)
6:         if (a = True) then
7:             c ⟵ StoreDataBC(Da)
8:             H_P ⟵ GenPointer(HashDa)
9:             for (S_j ∈ SubscribedList(P_k)) do
10:                Eth.send(H_P, S_j)
11:            end for
12:        end if
13:    end while
14: end function
```

Algorithm 1

(v) *ReadLocalData*(): function to retrieve data from the local buffer

(vi) *SendingIoTDeviceID*(): function to retrieve the ID of the sending IoT device

(vii) *StoreData*(): function to store data in the off-chain database

(viii) *GenPointer*(): function to generate pointer to location in the database

(ix) H: the Hash pointing to the location in the database

(x) *SubscribedList*(P): function to return the list of subscribers that subscribed to publisher P

(xi) *Eth*: reference to the blockchain client

(xii) C_j: set of j clients having the same subscriber

(xiii) *rec*(S): function that allows the client to retrieve the message sent by a subscriber S

(xiv) *Connect*(DB_P): function to connect to database DB_P

(xv) *fetch*(H_s): function to retrieve data corresponding to hash H_s

After describing the contracts, we specify in this section the behavior of the publisher node after receiving new data generated from the connected IoT devices. The steps that should occur are the following (1) store the data in the off-chain database (IPFS in our case), (2) generate the hash pointing to the location in the database, then (3) send that hash to every node on the blockchain that is subscribed to this publisher. All the transactions are sent using the blockchain client, which is an Ethereum client in our implementation. The publisher behavior algorithm is specified in Algorithm 1.

In the following, we will describe the behavior of a subscriber upon receiving a transaction sent by a publisher that it is subscribed to. Once a transaction is sent by a publisher, the blockchain client of the subscriber receives it and extracts the included hash. The subscriber first verifies the address of the publisher to make sure that it is not an

```
1: function Subscriber                                              ▷
2:     initializeSubscriber
3:     C_j ⟵ registeredclients
4:     while eventfromPublisher = True do
5:         H_s ⟵ Eth.receive()
6:         if ((H_s = 0) and (Eth.address(sender) == Eth.address(P_k))) then
7:             for c_j ∈ C_j do
8:                 send(H_s, c_j)
9:             end for
10:        end if
11:    end while
12: end function
```

Algorithm 2

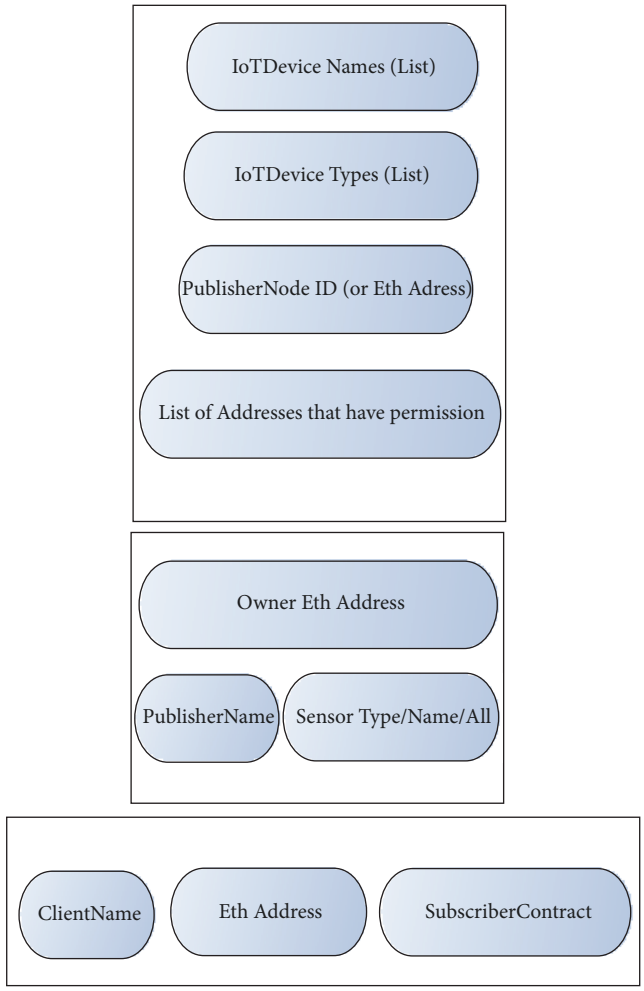


Figure 3: Proposed smart contracts: publisher contract, subscriber contract, and client contract.

error or a spam. If not, the action of the subscriber is different depending on whether the client is directly connected to the blockchain or not. If it is directly connected, this means that the client has deployed its own subscriber node therefore it receives the transaction directly through the blockchain client

```
1: function Client                                                  ▷
2:     initializeClient
3:     while (eventfromend − user = True) do
4:         H_s ⟵ rec(S)
5:         if (H_s = 0) then
6:             C ⟵ Connect(DB_P)
7:             Data ⟵ C.fetch(H_s);
8:         end if
9:     end while
10: end function
```

Algorithm 3

part of its subscriber node. Alternatively, the client may not have its own subscriber node and is therefore using another subscriber node to the blockchain along with other clients. This part implements the possibility of having several clients behind only one subscriber node handling the data on their behalf in the blockchain and forwarding it to all of them. The clients registered with the subscriber are part of the set S_j

The algorithm that shows this behavior is specified in Algorithm 2.

The third part describes the fetching of the data using the received hash in the second part. The client connects to the IPFS node whenever the end-user requests it, and uses the hash code to fetch the data generated by the IoT devices. This behavior is specified in Algorithm 3.

Figure 4 highlights the specific modules of the subscriber node and the publisher node. In the subscriber node, the Graphical User Interface (GUI) is the entry point for the end-user. The Ethereum client connects the node to the blockchain and the local database stores the received data (it works as a cache to reduce response time of future accesses).

It is also worthy to mention that one important step of the process is not actually visible in the Figure 4 that is the mining. Indeed, for each transaction taking place, the nodes on the blockchain (miners) shall validate the transaction as previousely explained. This increases the time to deliver a message from a publisher to a subscriber and it has also an important impact on the performance of the overall system in terms of response time.

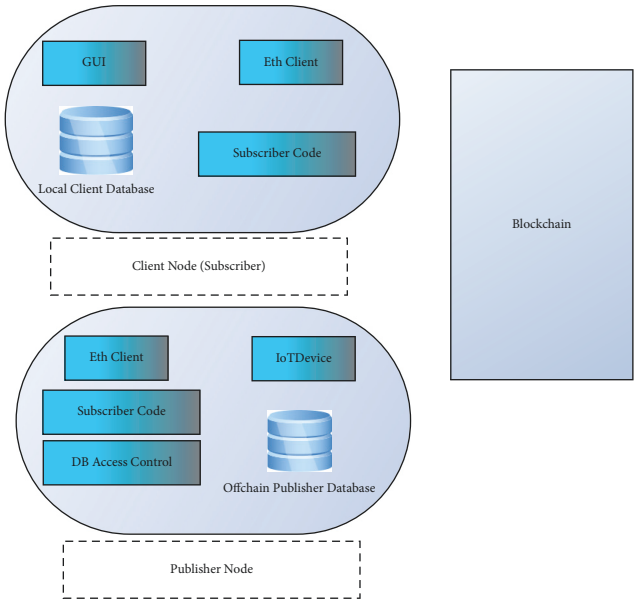


Figure 4: Internal structure of the client and subscriber nodes and publisher.

Figure 5 shows the sequence diagram of the proposed protocol that depicts the exchanged messages between the different components of the architecture. The previously presented algorithms specify the internal behaviors of the publisher and the subscriber nodes as well as the information that transit between them. The miner behavior corresponds to the mining process that happens each time a transaction is transmitted to the blockchain.

Table 1 describes the main functions used in the implementation of the proposed protocol. Each function is also described in this table. In Table 2, the main used variables that allow the protocol to function correctly are described. The variables presented in Table 2 constitute the input or output parameters to the functions described in Table 1.

5. Performance Analysis of the System

As previously mentioned, the mining process is an important part of the system and should be explicitly considered in the performance evaluation of the system mainly in terms of response time. Eventually, to optimize the performance of such a system, many factors should be taken into account. The main performance parameter is the delay of processing a transaction and therefore it is important to identify the parameters/factors that could have an important impact on this delay.

In this system, the transaction time obviously depends on the rate and size of the data; $t = f(r, s)$. This rate is important because it impacts the mining process. If the rate of data generated by IoT devices increases, the transaction time will also increase. The mining process depends also on the capability of a device processor to validate a transaction, so does the delay time [23]. The best approach to reduce the transaction time is to lower the rate and the size of the data. Unfortunately, this is not always possible. First, it is not always possible to control the CPU computing capacity since it comes with the device hardware and moreover it is not always possible to control the rate and the size of data transmitted by sensors (this also depends on the application logic). For example, monitoring in real time the ECG of an individual can be critical. For that, doctors must be

Table 1: Protocol functions and description.

Function	Description
RegisterPublisher	This function's role is to identify the publisher to the blockchain. A publisher registration consists of an ID and its address in the blockchain.
RegisterSubscriber	Similar to RegisterPublisher function, but for the subscriber nodes.
AddSensor	This function is available for the publisher nodes. In order to add sensors, a publisher should provide a type and an ID for the sensor.
RegisterToSensor	The Subscriber can check a publisher's added sensor, and register to one or all sensors, depending on ID or Type.
Notify	The notify function is critical, since it advertises new data, new publishers, and new sensors.
StoreData	When new data is generated, it is stored in the decentralized storage.
GenerateHash	After storage, a hash of the pointer to the location of the data in the storage is generated, and then broadcasted to interested subscribers using the Notify function.

Table 2: Protocol data types and description.

Data Type	Description
PubID	The publisher ID needed for registration, and used by subscriber to subscribe to, of type String.
PubAddress	The publisher address in the blockchain needed for registration, and used by subscriber to subscribe to, of type address.
SubID	The subscriber ID needed for registration, of type String.
SubAddress	The subscriber address in the blockchain needed for registration, of type address.
SensorInfo	The sensor type and ID, of type String.
HashPointer	The main data type transferred between publishers and subscribers, using the Notify function, can be of type base58.

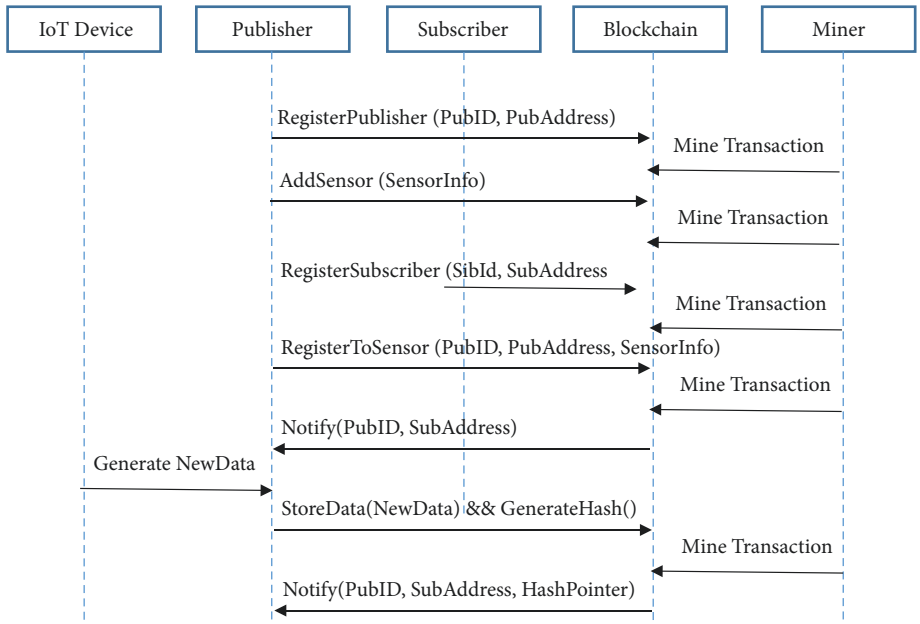


Figure 5: Sequence diagram of the Publisher-Subscriber protocol implemented on the blockchain.

able to interpret the received ECG data. In this case, heart measurements should be sampled and sent at high rate. This is not true for all types of physiological signals. For instance, it is not critical to measure body temperature at high frequency. Thus, changing the sampling rate of the ECG signal without a careful validation from doctors may not be good for an accurate interpretation of the data. Therefore, we consider in this work the mining time as the main performance metric to evaluate the performance of our proposed system:

Let N_d be the number of devices connected to a publisher.

Let r_d be the rate of IoT device data generation, and consequently the associated rate of generation of new blocks in the blockchain.

The total data rate generated by one publisher that has N_d associated IoT devices, assuming that all the devices have the same data rate r_d is equal to $N_d \times r_d$ sample of data per second.

Each new data will generate a new transaction which in turn will trigger the mining of a new block in the blockchain. In this case, the mining rate will depend on the block size N_t and the time to mine the generated block T_b. The block size depends itself on the number of transactions one block can fit; it varies indeed from one blockchain platform to another. As previously indicated, Bitcoin uses a fixed size for its block while Ethereum uses a variable size.

In this context, the mining rate will correspond to the fraction N_t/T_b bps. There are two possible cases; the first one is when the generated data rate from the publisher is lower than the mining rate and the second case is when it is higher. The maximum delay time for each case is formulated is the following:

Case 1: Max Delay = T_b if $N_d \times r_d \leq N_t/T_b$ Case 2: Max Delay = $N_b \times T_b$ where N_b is the number of necessary blocks to accommodate all transactions if $N_d \times r_d \geq N_t/T_b$.

N_b can be calculated supposing t=1s $N_b = (N_d \times r_d)/N_t$.

In the same way this can be generalized to all the generated data from all publishers. We obtain the following formulation:

$$\sum_{n=1}^{N_P} N_d^P i \times r_d^P i \geq \frac{N_t}{T_b} \quad (1)$$

In this formulation, we did not take into account the work and time taken by the subscriber and the publisher for their own operations. We consider indeed that these times are negligible comparing to the mining time which constitutes the most important part of the delay.

6. Implementation of the Solution

To implement our solution, we have chosen to use a framework called Embark [24]. This framework is based on the concept of decentralized applications (DApps) and was fitting our technical requirements. Embark framework implements the Ethereum blockchain, IPFS database, and Whisper protocol used to send messages between multiple DApps. We have chosen to use the GoEthereum (geth) client, along with SolidityC [25] language for smart contracts programming. HTML, Javascript and JQuery langages have been used to program the frontend and the GUI. We have successfully implemented our private blockchain as well as the notification smart contracts. We have considered two types of nodes, a laptop and a Raspberry Pi. We have successfully connected both of gateways to the blockchain. The Raspberry Pi permits to generate real data from its embedded sensors.

Technical Details. In this section, we discuss the settings of the system and the implementation details of the testbed.

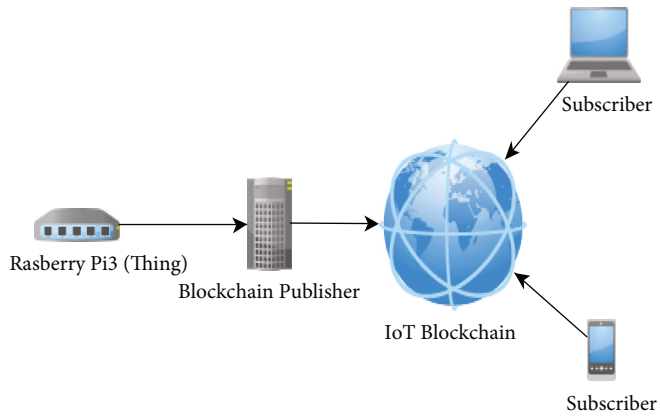


Figure 6: Testbed composed of a Raspberry Pi 3 as the publisher to the blockchain and a laptop as a subscriber to the blockchain.

Figure 6 shows our testbed that is composed of two nodes, a Raspberry Pi acting as a passive publisher (i.e., it is not a miner node) and the laptop acting as a subscriber as well as a miner for the blockchain. To validate the system architecture, we have implemented the scenario presented in Figure 5. The scenario involves first a publisher that registers to the system. The registration of the publisher suffices providing its name, since the Ethereum address is automatically fetched using the "msg.sender" function. Once the publisher is registered, The system identify the list of sensors provided by this particular publisher. For that, an advertising protocol is used (it is a core function in the proposed pub-submodel). The sensor is modeled in the publisher contract as a record of two variables, the sensor ID and the sensor type. Once the publisher and its sensors are registered to the blockchain, any subscriber connecting to the blockchain can discover the publisher and it sensors. In the general case, when a particular subscriber registers to the system, it is possible for it to discover all the available publishers via this particular subscriber as well as their provided sensing capabilities. It is then possible for a subscriber to subscribe to any of these publishers if it is authorized. If successfull, the address of the subscriber is added to the publisher's contract.

Once new data is generated from any of the sensors associated with a publisher, the gateway stores it in the IPFS storage and sends a broadcast message to the blockchain containing a hash pointing to the location where the data is stored. When the notification is received by a particular subscriber (in our testbed the laptop), its frontend module checks its contract that contains all the publishers' addresses to verify whether this publisher is among its associated publishers list. If the notification source address does exist in the list then the notification is accepted otherwise it is filtered.

The implementation of the system behavior was challenging since SolidityC was not completely mature, many features were not yet implemented at the implementation time. To verify that the GUI was working properly, we have performed several tests using the Embark dashboard.

The different steps of the system implementation and deployment are the following:

(i) Setting up the private blockchain using Geth.

(ii) Configuring the laptop and the Raspberry Pi and connecting them to the blockchain.

(iii) Programming the proposed smart contracts model using SolidityC.

(iv) Deploying the contracts on the blockchain.

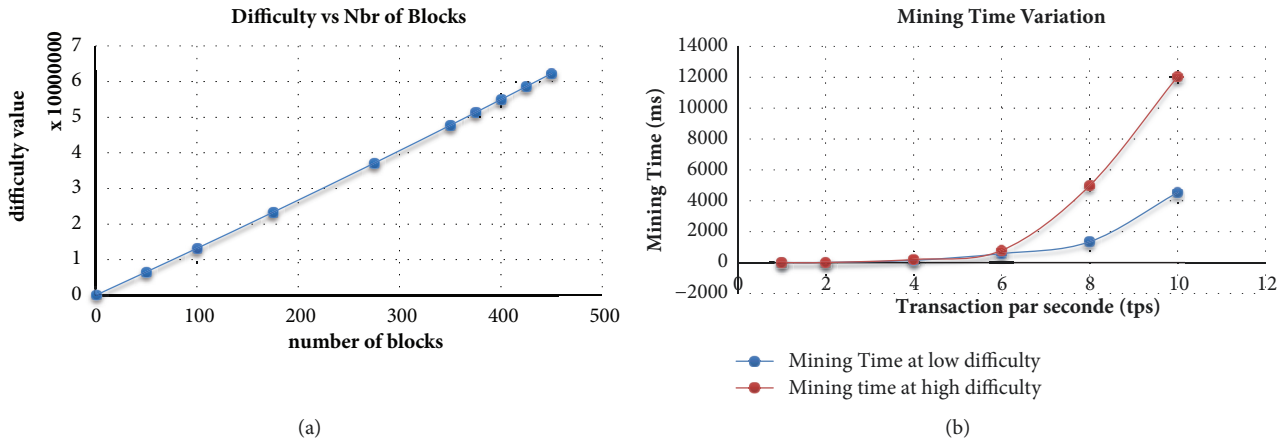
(v) Programming the subscriber's frontend (the HTML webpage).

(vi) Programming the backend (Javascript to link the smart contracts and the frontend)

(vii) Configuring IPFS off-chain database.

(viii) Connecting all the elements using Embark framework.

The implementation details of the different steps are described as follows: the first step to implement the system was to create a private blockchain to ease testing and experimentation. For that, we have used Geth (the command line interface for running a full Ethereum node implemented in Go) to create the Genesis file (the Genesis block is the first block in the blockchain), which is a Json file. After creating the Genesis file, we have specified some properties of the blockchain such as identity, rpcport, port, data directory and network Id. The next step was to test the blockchain by adding peers, connecting them to the blockchain, and sending Ethereum messages to make sure that the blockchain was working properly. Thanks are due to the Geth Javascript console and the Geth functions which allowed us to test the blockchain. Once the blockchain created, the following step was to create and deploy smart contracts. As previously stated, we have specified the contracts described in Section 5 and then used the SolidityC compiler to generate them. After successfully setting up the private blockchain and deploying the smart contracts, the backend was eventually complete. The next step was the deployment of GUI, i.e., the frontend of the DApp. The implemented frontend is composed of different files: html, Javascript and css files for the portal webpage and its interactions, along with several Embark configuration files. The visible part of the GUI is the portal webpage running on a webserver executed in the laptop. A webserver configuration file was also available in Embark.



(a) (b)

Figure 7: Difficulty variation versus number of blocks and time variation versus transactions rate for different difficulty values.

This environment was then used to deploy a blockchain storage GUI, the frontend and backend. Finally, we deployed the IPFS database and connected it to the private blockchain. All peers could then connect to the blockchain and store data in the IPFS off-chain database. This has completed the implementation and the deployment of the testbed.

Tests and Results. Before presenting the results of the tests, we will first present the key parameters of a blockchain:

(i) Number of transactions per block. It is calculated as the block size divided by the average transaction size. In fact, the number of transactions in a block in the Ethereum blockchain can reach 2200 transactions per block as a maximum value and 1050 transactions per block as a minimum value [26].

(ii) Transaction throughput which depends on two main values: the block size and the expected time interval between blocks.

(iii) Block time, mining time, or the time needed to validate a block.

(iv) Number of transactions per second (tps).

We have evaluated the variation of the mining time against the blockchain scale. This was achieved by varying the number of transactions per second (tps).

The difficulty level parameter of the Ethereum blockchain (i.e., level of difficulty to mine blocks) [26] is directly related to the mining time. In fact, the difficulty is a scalar value corresponding to the difficulty level applied during the nonce discovery of the processed block. It defines the mining target, which can be calculated from the previous blocks difficulty level and the timestamp. The higher the difficulty is, the statistically the more calculations a miner must perform to discover a valid block. This value is used to control the block generation time of a blockchain, keeping the block generation frequency within a target range. In our testbed, we have kept this value intentionally low to avoid waiting too long during tests. Indeed, since the discovery of a valid block is required to execute a transaction on the blockchain, the overall operation can take a lot of time. We have made our tests when the difficulty value was low, and after a certain time, we have conducted other tests when the difficulty value has increased to high values. In Bitcoin, the average expected system throughput value is 1.75 tps. It is worth noting that the miner, actually the laptop has the following technical configuration in this testbed: Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz, 8.00GB RAM, 64 bit OS.

Figure 7(a) shows the variation of the difficulty value versus the number of blocks. Actually, the difficulty increases linearly with the number of blocks which is a property known for the blockchain however Figure 7(b) shows that the mining time increaseing exponentially with the rate of the transaction (i.e., number of transactions per second) which is also inherent to the blockchain behavior since more transactions the blockchain needs to process per second higher the processing capacity is required from the blockchain Miners. The following Figure highlights also the difficulty value increases, the mining time increases significantly which is also a characteristic of the blockchain that impacts our solution. Therefore there is need to find the right tradeoff between the difficulty level and the target response time of the system which will be perceived by the end-users.

7. Conclusion and Future Works

The objective of this work was to consider the possibility of using blockchain technology in the area of IoT data access protection with a possible application in the eHealth area with the protection of personal medical information collected from medical sensors and environmental sensors in smart homes. We have proposed an architecture of a solution designed for that purpose that is based on contracts between providers and consumers of data. To cope with the size of the data to store, we have proposed to associate the blockchain with an off-chain database. The block contains the main contract information as well as reference to where the complete data is stored. We have also discussed and presented the performance of such a system and the parameters that may have an impact on the time to process the transactions.

We have implemented the system in a testbed and conducted some tests to validate the behavior of the system components. We have shown that existing technologies have permitted to implement the proposed architecture. Finally, we have performed some performance measurement of the system to highlight how the system response time (mining time) varies against the rate of the transactions that is an important factor to consider when deploying such a system in eHealth realm. In our future work, we aim to extend the system to work on a public blockchain and conduct large scale evaluations.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the University of Evry Val dEssonne and by the Lebanese University and CNRS Lebanon. Part of this work was also conducted in the frame of the PHC CEDRE Project N37319SK. The authors also thank their colleague Dr. Massum Hasan for the early discussions on the topic.

References

[1] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, "World of empowered IoT users," in *Proceedings of the 1st IEEE International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pp. 13–24, Berlin, Germany, April 2016.

[2] N. Nakamoto, "A Peer-to-Peer Electronic Cash System," 2008, https://bitcoin.org/bitcoin.pdf.

[3] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[4] Nick. Szabo, The Idea of Smart Contracts, http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html.

[5] N. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, http://bitcoin.org/bitcoin.pdf.

[6] M. E. Peck, "Blockchains: How they work and why they'll change the world," *IEEE Spectrum*, vol. 54, no. 10, pp. 26–35, 2017.

[7] A. Judmayer, N. Stifter, K. Krombholz, and E. Weippl, "Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms," *Synthesis Lectures on Information Security, Privacy, and Trust*, vol. 9, no. 1, pp. 1–123, 2017.

[8] "Ethereum homestead documentation," http://www.ethdocs.org/en/latest/.

[9] N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)," in *Principles of Security and Trust*, vol. 10204 of *Lecture Notes in Computer Science*, pp. 164–186, Springer, Berlin, Germany, 2017.

[10] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *Proceedings of the 18th IEEE International Conference on e-Health Networking, Applications and Services, Healthcom 2016*, Munich, Germany, September 2016.

[11] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proceedings of the 2nd International Conference on Open and Big Data, OBD '16*, pp. 25–30, Vienna, Austria, August 2016.

[12] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[13] J. Zhang, N. Xue, and X. Huang, "A Secure System for Pervasive Social Network-Based Healthcare," *IEEE Access*, vol. 4, pp. 9239–9250, 2016.

[14] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proceedings of the IEEE Security and Privacy Workshops, SPW 2015*, pp. 180–184, San Jose, Calif, USA, May 2015.

[15] M. Y. Jung and J. W. Jang, "Data management and searching system and method to provide increased security for IoT platform," in *Proceedings of the 2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 873–878, Jeju, South Korea, October 2017.

[16] C.-F. Liao, S.-W. Bao, C.-J. Cheng, and K. Chen, "On design issues and architectural styles for blockchain-driven IoT services," in *Proceedings of the 4th IEEE International Conference on Consumer Electronics - Taiwan, ICCE-TW 2017*, pp. 351-352, Taipei, Taiwan, June 2017.

[17] F. Angeletti, I. Chatzigiannakis, and A. Vitaletti, "The role of blockchain and IoT in recruiting participants for digital clinical trials," in *Proceedings of the 25th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2017*, Split, Croatia, September 2017.

[18] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, pp. 618–623, Kona, Hawaii, USA, March 2017.

[19] N. Rifi, N. Agoulmine, N. Chendeb Taher, and E. Rachkidi, "Towards using blockchain technology for data access management," in *Proceedings of the IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB Conference)*, 2017.

[20] "Mining in Bitcoin," https://en.bitcoin.it/wiki/Mining.

[21] IPFS, "Content Addressed, Versioned, P2P File System," https://ipfs.io/docs/.

[22] "Ethereum whitepaper," https://github.com/ethereum/wiki/wiki/White-Paper.

[23] J. A. Dev, "Bitcoin mining acceleration and performance quantification," in *Proceedings of the 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering, CCECE 2014*, Toronto, Canada, May 2014.

[24] "Embark framework," https://github.com/iurimatias/embark-framework.

[25] Solidity Introduction to Smart Contracts, http://solidity.readthedocs.io/en/develop/introduction-to-smartcontracts.html.

[26] "Ethereum charts and statistics," https://etherscan.io/charts.