Management of Medical Information in IoT Using Block Chain

Pengshu

April 5, 2018

Medical devices in IoT generates sensitive information private to patients, such data requires permission management, and the content of data requires hiding. We use block chain technology to manage the permission to data, and apply one-time-key cryptography to protect the data privacy of patients. Upon such convenient and secure way of accessing medical data, we propose methods to break the data barrier between medical institutes and ethically share patient data for medical research.

1 The Problem Statement

It is difficult for patients in eastern Asia, especially China, to access their own medical data. Also, there is data barrier between medical institutes. Such nonavailability of medical data causes redundant examinations, jeopardizes the relationship between patients and medical workers, and is developing into a social problem.

In the light of IoT technology, it is possible that various medical devices joins a network to provide convenient medical data access for patients and medical institutes, immensely improving the data availability.

Note a point to bear in mind is, medical data contains sensitive personal information about patients, not everyone should have the permission to view any piece of medical data on the network. Medical data of a patient should only be accessible under the request of the patient himself or responsible medical workers. Block chain based solution to manage permission to medical databases has been put forward by Azaria et al.[1], and can be extended to managing data generated by IoT medical devices.

Another problem is leakage of medical data, which may occur during an attack on medical database or IoT medical devices. Statistics have shown that these event are not uncommon. IoT medical devices are particularly vulnerable due to limited computing power. Block chain itself is not enough for ensuring data security at the lower level of storage provider, as is clarified by Azaria et al.[1]. A solution which does not requires extensive computing power while keeps the data safe is favorable for IoT medical devices.

2 Existing Method of General Medical Record Management using Block Chain

Azaria et al.[1]implemented a system using block chain and smart contract to manage the access and permission off-chain medical data. In their model, the central identities involved are Provider and Patient, who collaborate to manage the medical data of Patient. Provider provides the physical storage space for the medical data of Patient, while Patient makes queries to the Provider in order to retrieve his/her medical data.

A private block chain is deployed in the system, Provider and Patient acts as nodes in the block chain network. There interaction between Provider and Patient on chain are achieved my manipulating 3 types of smart contracts.

1. The issuing of virtual ID to Providers and Patients is managed by **Registrar Contract**

2. The establishment of collaboration between Provider and Patient, and access verification to a certain piece of medical data is managed by **Provider-Patient-Relationship(PPR) Contract**

3. The activities of updating medical data, and querying medical data, is recorded and managed by **Summary Contract**

The medical data of Patient is stored in offchain database managed by Provider. Updating the medical record of Patient works as follows:

- Update of medical data of a particular Patient is received by Provider, say the doctor uploads the latest medication of Patient to Provider.

- Provider updates the entry for Patient in the offchain database.

- Provider resolves the Provider-Patient relationship onchain, and posts transactions on block chain to update the Summary Contract and PPR contract.

- The Summary Contract fires a signal to Patient, notifying the update.

Querying the medical record of Patient works as follows:

- Patient send a digitally signed query request to Provider, using an offchain communication channel.

- Provider checks onchain the accessibility of the sender of query request.

- If the sender of the query request has the permission level to access the query content, Provider retrieves the corresponding query content from offchain database, and return the content to Patient.

A few remarks to make:

NO medical data is stored on block chain in this model.

The block chain only manages the accessibility of users to medical data, , and the Patient-Provider-Relationship, in a highly secure way.

The actual medical data is stored offchain, the data format is only dependent on the structure of offchain database, therefore the content and formatting is not restricted.

The block chain CANNOT deal with information leakage from user end. Both Patient and Provider are responsible for the data admin in their local storage, and their offchain communication.

3 Secure Medical Data Generation Using Two-Key-Encryption

3.1 Cryptographic Keys

In our model, each Patient has a **Emergency Key**. **Emergency Key** is used to encrypt and decrypt a copy of medical data needed in emergency, including information about blood type, allergic history, medication history, etc.

Each Patient can generate a pair of **Temporary Public Key**, **Temporary Secret Key** every time when using a medical device. **Temporary Public Key** is given to desired medical device to encrypt data generated, **Temporary Secret Key** is kept by Patient or responsible medical workers to later decrypt the encrypted data.

We will define the generation of keys formally as below.

Each Patient should have a unique Patient ID in order to use this system. The Patient ID should normally be the social security number, citizen ID number, or medical insurance number, which are generated outside the system.

Definition 3.1. $EmergencyKeyGen(PatientID \| randomness)$

EmergencyKeyGen() takes in the PatientID concatenated by a random number, returns a new **Emergency Key** based on these two arguments.

Definition 3.2. $TempKeyPairGen(PatientID \| randomness)$

TempKeyGen() takes takes in the PatientID concatenated by a random number, returns 2 keys, one is **Temporary Public Key** for encryption, the other is **Temporary Secret Key** for decryption.

The secret key generated by TempKeyPairGen() should be kept secret by Patient, the public key generated by TempKeyPairGen() should be public, and given to the medical device to encrypt new data generated.

3.2 Data Encryption and Decryption

For the convenience of our discussion, we define a stub function which returns the plain medical data.

Definition 3.3. DataGen()
DataGen() takes no argument and returns the plain data generated by medical device.

Now we are ready to discuss how we secure emergency data and general data upon generation, how they are stored and decrypted.

```
Data: PatientID, StorageAddr
emdata=DataGen();
/* Generate Emergency Key                                        */
emk=EmergencyKeyGen(PatientID‖rand());
/* Encrypt emdata using secret key encryption                    */
encode=s_encrypt(emdata,emk);
/* Send encoded data to data storage                             */
send_emergent(encode,StorageAddr,PatientID);
some_time_has_passed();
some_emergency_happens();
/* Retrieve Encrypted Emergency Data With Reference to Patient ID
   */
encode_data=retrieve_emergent(StorageAddr,PatientID);
/* Decrypt emergency data using Emergency Key                    */
data=s_decrypt(emk,encode_data);
```

Algorithm 1: How to secure emergency data

Patient uses **Emergency Key** to both encrypt and decrypt his/her emergency data. The encrypted emergency data is stored in storage device. The encrypted emergency data is retrieved by referring to Patient ID.

```
Data: PatientID, StorageAddr
/* Generate Temporary Key Pair, this must be done before
   generating medical data                                  */
pk1,sk1=TempKeyPairGen(PatientID‖rand());
/* Get data from device                                     */
data=DataGen();
/* Encrypt data using public key encryption                 */
encode=p_encrypt(emdata,pk1);
/* Send encoded data to data storage, get reference number for
   this piece of data                                       */
ref1=send_general(encode,StorageAddr);
/* Make another medical measurement, we want to use a new one time
   key                                                      */
pk2,sk2=TempKeyPairGen(PatientID‖rand());
data_prime=DataGen();
encode_prime=p_encrypt(emdata,pk2);
ref2=send_general(encode_prime,StorageAddr);
some_time_has_passed();
/* Retrieve the first piece of data With Reference to Reference
   Number                                                   */
encode_data=retrieve_general(StorageAddr,ref1);
/* Decrypt the first piece of data using the first Temporary
   secret key                                               */
data=p_decrypt(sk1,encode_data);
/* Retrieve the second piece of data                        */
encode_data=retrieve_general(StorageAddr,ref1);
data=p_decrypt(sk1,encode_data);
```

Algorithm 2: How to secure general data

Every time when Patient do a new measurement using some medical device, he/she must specify in advance what public key to use. The data is encrypted using public key. The encrypted data is stored in storage device. The encrypted data is retrieved to by referring a reference number dedicated to the data, and inside retrieve_general undergoes the permission control process using block chain, which is discussed in the next section. The data is decrypted using secret key.

4 Lightning Permission Management Using Block Chain

We adopt the methodology of MedRec[1]to perform permission control on encrypted medical data. As the data are stored distributively on small devices, the permission verification is automated and simplified, so as to accommodate the limited computing power and network traffic of IoT devices.

How to do this in detail, hmmm... need to think about it.

5 Data Sharing

Given the user end encryption and permission management is properly set up, sharing the data of a patient is straight forward. The Patient simply publish his the ref number of his data and the corresponding temporary secret key for decryption on block chain, then signal the smart contract to and make public the accessibility of that piece of data.

How to make public the accessibility of that piece of data, hmmm... need to study MedRec further more.

References

[1] 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)*, pages 25–30, Aug 2016.