Healthcare Blockchain

In the following blockchain implementation, there exist 4 classes corresponding to the 4 different actors of the blockchain: ***Minister, Doctor, Patient*** and ***Miner***. Every class has peculiarities for the different agents and every instantiation of a class can be thought as of user account by which the agent can do the allowed operations related to the class from which it instantiates. For instance, if we instantiate the class ***Patient*** in the following way:

```
patient1 = Patient('Mr. Black')
```

Then the patient1 memory address in the local machine will be the address used by the blockchain to do transactions with that particular patient. This allowed us to think to every patient (patient1, patient2 and so on) as real user account from which the patients do operations. The same applies to the other classes.

The blockchain has 3 types of transactions: ***diagnosis***, ***prescription*** and ***authorization***. Here an example for each of them. In the first case the doctor makes a diagnosis, and when the block is mined the illness1 is appended to the history of illnesses of that particular patient. In the second case, instead, we have a prescription and in the third case the authorization. In this last case, after the block is mined, the authorization is inserted into the variable *authorization* of that particular doctor.

- **type**: 'diagnosis',
  sender: <unique_blockchain.blockchain.Doctor object at 0x0000026A76FAAD30>,
  recipient: <unique_blockchain.blockchain.Patient object at 0x0000026A76FD5438>,
  illness: illness1
  fee: 0.2
- **type**: 'prescription',
  sender: <unique_blockchain.blockchain.Doctor object at 0x0000026A76FAAD30>,
  recipient: <unique_blockchain.blockchain.Patient object at 0x0000026A76FD5438>,
  prescription: medicine3
  fee: 0.2
- **type**: 'authorization',
  sender: <unique_blockchain.blockchain.Minister object at 0x0000026A5FFB7320>,
  recipient: <unique_blockchain.blockchain.Doctor object at 0x0000026A76FAAD30>,
  authorization:
  b"L\xd8\xdfCy/\xea\xf7}\x9f\xb2q\xec\x12Z2)(F*\xee\xf7\xdb\xb4\xda\n\xb4\x0f”
  fee: 0.2

After the mining of the block the fee is assigned to the miner who mined the block in the least time. Since it was not possible to run in parallel, we decided to let do the proof of work to each miner, computing the time for each one and giving the fee only to the fastest one. This was done only for simulation purposes.

Writing Permissions

The idea behind the implementation is that only doctor can do diagnosis or prescriptions to patients, this means that only those who “open a doctor account”. If this is the rule, the problem that comes out is that potentially everyone can open a doctor account from which can sends diagnosis to patients. For this reason, it was necessary to create an authentication mechanism by which only doctor verified had writing permissions. The class Minister is useful to do that. Every time a doctor is enabled by the Medical association of the country in which he works, the Minister of Health issues an authorization

to them in order for them to make transactions. Technically speaking the authorization is based on the digital signature mechanism. In particular, the unique address of the doctor is signed by the Minister using the private key of the Minister and the resulting encryption is the authorization which is then stored in a variable called “authorization” which is specific of the Doctor class. The creation of the authorization is a transaction as well, where the sender is the Minister of Health, the recipient is the doctor and the object are the authorization. Every time a doctor will do a diagnosis or a prescription, the miner, before mining the block, will try to decrypt the authorization of the doctor using the public key of the Minister of Health. If this goes on, then the authorization is valid, otherwise either the authorization is fake or that doctor still does not have an authorization.

Checking incompatibilities

The other improvement that the blockchain brings is the possibility to check automatically to incompatibilities among illnesses had by the patient and potential side effects of drugs given by the doctor. In particular, after the check of the doctor’s authorization, the miner checks also that, in case of prescription, the patient did not have an illness which is incompatible with the drug given. To do that, the instantiation of the Minister of Health has a variable called *incompatibilities* storing, for each drug the illnesses which can cause side effects. The miner checks that in an automatized way, without accessing directly the data of the patient. In this way, the Minister is the only agent having all the incompatibilities and new ones can be added only by the Minister.

Reading Permissions

Since Doctors need to look at the illness history of each patient, they need to have an access to them, but at the same time this has to be done just for a single time, preventing the doctor from having access to the complete history of the patient also in the feature. For this reason, we implemented a temporary key mechanism and the idea is the following: every patient has a private and related public key. If the doctor wants to access the history of the patient, he needs the private key of the patient. The blockchain gets the private key and extrapolate a public key from that, then it retrieves the real public key from the variable of that patient in an automatized way and finally compare the real public key and the one extrapolated by the private passed by the doctor. Once the doctor has access to the history of the patient the private and the public keys of the patient are refreshed. Therefore, if the Doctor tries to access again with the previous private key, the system will throw an error because the real private and so, the real public key has changed.