Designing a distributed personal medical system

A. Alochukwu[†], V. Kubalasa[⋆], P. Mokoena[⋆],
E. Mwenitete[‡], and M. Olusanya[*]

[†]University of Johannesburg
[⋆]University of the Witwatersrand
[‡]African Institute for Mathematical Sciences
[*]Sol Platje University

Abstract

The abstract comes here

1 Introduction

This project begins with a brief survey of centralised versus distributed systems, emphasising the local thinking needed to design a distributed protocol which achieves some global property. It then considers the features and functionality desired of a distributed personal medical system, emphasising those not achievable in a conventional system (for instance exploitation of data analytics). Ethics, accountability, and intrusion are important considerations that participants may not be used to addressing. A system is designed which incorporates the desired features. Coding is not required, but reasoning about correctness of behaviour of our design is. A distributed system will be thought of as a 'data structure' i.e. as consisting of state on which certain operations are defined, like 'Update a record', and 'Query a record'. We must decide what 'state' the system has in order to be able to express its operations accurately. The first step is to understand what exactly is a 'data structure' and how to describe it using discrete mathematics in the Z notation.

2 Problem Statement

Question: Can we design a distributed (i.e. de-centralised) system to manage personal medical history?

People expect to be, and are in normal years, far more mobile than ever before. Mobile phones and the web provide the communications and information they need, but national border control, personal ID and health systems remain resiliently conventional in spite of being essential for travel. By comparison currency has been potentially liberated, with the advent of Bitcoin. Can we do the same for personal health records?

Suppose you're stung by a jellyfish while on holiday in northern Queensland, break a leg skiing in the Alps, or catch a virus whilst working in Wuhan, immediate treatment requires knowledge of your medical history, allergies and insurance. Record books are unreliable, easy to lose and to keep with you at all times.

1. Is there some distributed system which allows access to your personal medical information from anywhere in the world?
2. What functionality should it support?
3. What are the ethical requirements of such a design?

3 Formulating PMS as an Abstract Data Type

In this section, we use a Z schema to model/ formulate the system as an ADT. Z provides a way of structuring discrete changes in a dynamical system. We refer to [1] for a summary of Z.

3.1 System

The following key questions will serve as a guide in designing a distributive system for personal health records.

1. What is the state of our system?
2. What operations does the system support?
3. Does the system maintain privacy?

The answers to the above questions will help us in specifying a discrete dynamical system as an Abstract Data Type (ADT).

For the system under consideration, the state schema for a particular individual consists of 2 observables that are also sub schemas, namely the Personal Database (PDB) and the medical History (Med. Hist)

3.1.1 State of an Individual

```
┌─ State ──────────────────────────────
│ PDB
│ MedHistory
└──────────────────────────────────────
```

PDB: This schema also consists of 5 observables that happen to be sub schemas, namely the: Personal, Medical, Contact, Financial and Travel.

```
── PDB ──────────────────────────────
│ Personal
│ Contact
│ Medical
│ Financial
│ Travel
└────────────────────────────────────
```

Personal: This schema contains all the individual patient's personal details as observable, such as their first name, last name, other names (if any), date of birth, unique ID (referring to their national unique ID), gender, nationality and their password to access their medical state.

```
── Personal ─────────────────────────
│ surname : Text
│ firstname : Text
│ othernames : Text
│ dob : Year/Month/Day
│ gender : LGBTQI+
│ uniqueid : Digit*
│ nationality : Text
│ password : Alphanumeric*
└────────────────────────────────────
```

Contact: Here we have a schema that contains all the patient's important contact details and next of kin's details. These include: phone number, work number, home number, email address, residential address, permanent home address, postal address, next of kin's name, relationship with next of kin, cell number of next of kin and address of next of kin.

```
── Contact ──────────────────────────
│ cellnumber : Digit*
│ homenumber : Digit*
│ officenumber : Digit*
│ emailaddress : Varcharacter
│ residentialaddress : Textnumeric
│ permanenthomeaddress : Textnumeric
│ employer : Text
│ employeraddress : Text
│ postaladdress : Textnumeric
│ nextofkin : Text
│ relationshipwithnextofkin : Text
│ phonenumberofnextofkin : Digit*
│ addressofnextofkin : Text
│ emailaddressofnextofkin : Varcharacter
└────────────────────────────────────
```

Medical: We now have all the individual's medical details recorded here as observables. The Medical comprises of blood type, genome, allergies and immunities.

Medical
bloodtype : *Alphanumeric*
genotype : *text*
bloodpressure : *Digit**
weight : *Digit**
allergies : *Text*
genome : *DNASeq*

Financial: This schema has the individual's medical finances and includes their medical scheme(if any), medical insurance name, medical aid number, medical aid plan and whoever is responsible for their account if they do not have a medical scheme.

Financial
medicalaid : *Text*
medicalplan : *Text*
medicalaidno : *Digit**
contactpersonforaccount : *Text*

Travel: The travel schema consist of a schema which is a sequence of data where the individual has traveled and thus contains all information related to that. The schema has 6 observables which are the date of travel, duration, destination, contact number and their purpose/reason for travel.

Travel
origin : *Text*
traveldate : *Year/Month/Day*
duration : *Alphanumeric*
destination : *Text*
purposeoftravel : *Text*
contactdetails : *Alphanumeric*

Med. History: This schema being part of the state, is given by a variable, say *mh*, which is simply a sequence of other schemas being medical events (seq Event) that occur over time and are recorded/appended to an individuals medical history, which is then attainable by those authorized in times of need.

MedHistory
mh : seq *Event*

Event: The event schema is also partitioned into the medical schema, the description schema and the financial schema. These schemas essentially contain the respective details of a specific medical event.

Event

Medical
Description
Financial
Referral

Medical: We record all the medical data associated with a specific event, which include the medical institution the individual had to go to, the medical contact number for the event and the medication/drugs prescribed to the individual for this event.

Medical

medicalInst : *Text*
medication/*operationrequired* : *Alphanumeric*
prescription : *Alphanumeric*
immunity : *Text*
localpatientcontact(*changes*) : *Alphanumeric*

Description: This schema contains the specifics of the incident. The observables required are where (describes where the incident had occurred in coordinate form), when (the date the incident took at place), what (the actually narrative of the event), diagnosis (what medical experts claim is wrong), referral (where the individual is to be referred to due to the event) and severity (a scale to measure the intensity of the event).

Description

whathappened : *Text*
place : *Text*
time : *Digit**
severity : *Text*

Financial: A schema with the financials linked with a specific event and the required information being the medical costs, the medical aid (if any) and the person liable for fees if individual does not have any medical scheme.

Financial

estimatecost : *Currency*/*Number*

Referral: This is the schema that allows an individual to give referral for consultation, review or for further action. A medical doctor or health practitioners refer patient to a senior medical specialist.

Referral

medicalinstitution : *Name*

3.1.2 System State

In this section, we outline the different schemas that are relevant to our design of a distributive system. We begin with a schema that captures the database of all registered individuals.

Reg: This schema with the observable *reg*, is a database consisting of everyone's personal details.

$$\begin{array}{|l}\hline \textit{Reg} \\ reg : \mathbb{P}\, Personal \\ \hline\end{array}$$

Allowed: Power set of all individuals who are allowed to access the person's data that is qualified medical doctors, health practitioners,etc. The criteria to be allowed is to have a medical license or relevant board certificate that will be digital verified.

$$\begin{array}{|l}\hline \textit{Allowed} \\ a : \mathbb{P}\, Individual \\ \hline\end{array}$$

The following schema is the state of our proposed distributed personal medical system that stores the required information for all registered users as well as deceased individuals.

SState: This is the state of our medical system and it has the database of all the registered individuals and all those that have passed on. Where the observable old refers to a data base that stores all the previously stored and deceased individuals medical history under users. The state system thus changes as people die or as we register new individual to the data base via specific operations.

$$\begin{array}{|l}\hline \textit{SState} \\ db : Reg \rightarrow State \\ old : Users \rightarrow State \\ Allowed \\ \hline\end{array}$$

3.1.3 System Operations

In this section, we outline different operations that our proposed system can support. These operations are structured using Z-schema. The operation includes but not limited to the following

Regnew: This schema allows us to add an individual ($p?$) to the data base and inevitable changes the PDB, hence there will be a change in system state. The registered list post addition (reg') is the union of the registered list prior (reg) with the addition of the new individual.

$$\begin{array}{|l}
\textit{Regnew} \\ \hline
\Delta Reg \\
p? : Personal \\ \hline
reg' = reg \bigcup \{p?\} \\ \hline
\end{array}$$

Update: To update one's personal details, we would need to operate on the PDB. There's a change in the system state due to this, our inputs being the individual's old information ($p?$), the new information ($new?$) and the password required to access the PDB ($pw?$). The password is checked to see if it correlates to the individual's selected password, before allowing access. Once that is done, we ensure the database of deceased remains unchanged. To update the new information, we use the operation override, where we set the updated database (db') to be the old database updates with the new information for a specific individual.

$$\begin{array}{|l}
\textit{Update} \\ \hline
\Delta SState \\
p? : Personal \\
pw? : Alphanumeric \\
new? : Reg \\ \hline
pw? = p?[pw] \\
old' = old \\
db' = db \bigoplus \{new? \to dp(p?)\} \\
reg' = (reg \backslash p?) \bigcup \{new?\} \\ \hline
\end{array}$$

Accident: This is the schema that allows us to record an accident/event that occurs. The inputs of this schema are the Individual involved in the accident ($p?$) and the event details that have occurred ($new?$). Like before, during an event, we assume no one dies and so the database of those who have passed does not change. Their medical history does change, and so we need to update this. The individuals old medical history is concatenated with the new event details, where no one else state has changed in the system.

$$\begin{array}{|l}
\textit{Accident} \\ \hline
\Delta SState \\
p? : Reg \\
new? : Event \\ \hline
old' = old \\
mh(p?) = mh(p) +\!\!+ [new?] \\
\forall\, q \neq p?\ mh'(q) = mh(q) \\ \hline
\end{array}$$

Get-Information (Read): This operation inputs ($p?$) a registered/previously registered individual personal information and only allows an individual ($q?$)

from the allowed schema to read or access their personal database and medical history.

$$
\begin{array}{|l}
\textit{Read} \\
\hline
\Delta \mathit{SState} \\
p? : \mathit{Reg} \mid \mathit{Users} \\
q? : \mathit{Individual} \\
d! : \mathit{State} \\
\hline
d! = dp[p?] \mid old[p?] \\
db' = db \\
old' = old \\
q? \in \mathit{Allowed} \\
\hline
\end{array}
$$

Death (Remove): When an individual is involved in an event that results in his death, the operation inputs the person's personal information, removes the deceased form the database of registered users and stores their database into the old database.

$$
\begin{array}{|l}
\textit{Remove} \\
\hline
\Delta \mathit{SState} \\
p? : \mathit{Reg} \\
\hline
db' = db \setminus \{p? \longmapsto db(p?)\} \\
old' = old \bigcup \{p? \longmapsto db(p?)\} \\
\hline
\end{array}
$$

4 System Design

To have a distributed system, we make use of blockchain (see [2] for more details).

4.1 An overview of Blockchain

According to www.euro money.com, Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.

This is a technology used extensively by cryptocurrency. This technology uses what we call a hash function which takes an unbounded string of data to a string of length N/finite (32 in the case of bitcoins).

We want to connect our successive medical events via blockchain, this is done by mining. The way we keep our data secure is by SHA (secure hash algorithm) , which is a algorithm that sets out the hash digits in a certain sequence.

We want to also ensure that only authorised personal are allowed to access and update the data in the system, we would only allow full nodes of the medical network to do so.

We will have the events and the PDB in each block, and blockchain is a connection of blocks and thus these sequence of events

4.2 Medical Blockchain

In this section, we give the state of a medical blockchain. To do this, we first consider the outlined key questions before giving a detailed description of the relevant schemas.

- What goes into a block? PDB and Events (Medical events)
- Block chain ?
- Who adds to Blockchain (Miners) ?

Block: The block schema contains an observable its (items) which are essentially a sequence of the data that is meant to be stored in this new block, namely being PDBs and events. The other observables are you left and right hash required for this nee block. As we may know by now, blocks can only contain a certain amount of data, and so we need to ensure that the items contained lie within the limit k. This threshold k depends on the system's implementation and how much space each PDB and each event consume. Once that has been calculated, we can decipher how many items a block can contain.

We now hash this new block containing the medical data and using the previous hash information.

With the previous block's hash being undefined (as shown by the perpendicular sign) we would require the previous blockchain before this operation and this is done by the add block schema.

Block

$its : \mathrm{seq}\, PDB/Events$
$L, R : \mathbb{B}^*$

$\#its = k$
$R = hash(its, L)$
$L = \perp$

Assume Initial Block (Genesis Block) is empty

Blockchain

$bc : \mathrm{seq}\, Blocks$

4.3 Medical Blockchain Operations

In this section, we use Z-notation to describe the different operations for our medical blockchain.

Add Block: This schema essentially allows us to add a block to the blockchain and thus store more medical data. It will change the blockchain as such and the observable input therefore being a block. We have our new block chain being the concatenation of our old blockchain with the added block. By doing so, we have link the previous block's hash information to the new block and compute the new block's hash information, containing all prior hash information.

$$\begin{array}{|l}
\hline
\textit{AddBlock} \\
\Delta \textit{Blockchain} \\
b? : \textit{Block} \\
\hline
bc' = bc \mathbin{+\!\!+} [b?] \\
b?[L] = \textit{lastbc}[R] \\
b?[R] = \textit{hash}(b?[L], b?[\textit{its}]) \\
\hline
\end{array}$$

Access: This schema now grants the access to whomever is authorised to do so (which will be the full nodes). These full nodes would be medical practitioners and legal entities. The inputs of this schema would be the full nodes ($w?$), the individuals whose data is being accessed ($p?$) and the output is thus the state of that specific individual ($r!$), where we can access the PDB of the individual and their sequence of events.

We know that by accessing such information, the blockchain remains unchanged . Those attempting to access this data must fall under a set of those allowed to access the blockchain, once granted access, they will be able to see the individual's data in the blockchain.

$$\begin{array}{|l}
\hline
\textit{Access} \\
\Delta \textit{Blockchain} \\
w? : \textit{Individual} \\
p? : \textit{Reg} \\
r! : \textit{State} \\
\hline
bc' = bc \\
w? \epsilon B \subseteq \textit{Allowed} \\
r! = bc[p?] \\
\hline
\end{array}$$

5 Future Work

We outline below some aspects of our system that needs improvement.

1. **Privacy:** So far we have a password that deals with the data privacy of an individual, however, since we want our system to be forward looking, we aim to include modern and more secure ways of allowing only the user to access their personal information such as liveness check, digital signatures, facial recognition, digital signatures, etc.

2. **Operations:** We will add more necessary operations to the system such as an error message popping up when an error is detected in the system

References

[1] The *Z* Notation: A Reference Manual.J.M.Spivey, Prentice-Hall, 2001.onlinebooks.library.upenn.edu/webbin/book/lookupid?key=olbp69629

[2] Distributed Ledger Technology: beyond block chain.A report by the UK Government Chief Scientific Adviser.OGL, Crown copyright, 88 pages, 2016. a

[3] Mathematical Underpinning of Analytics: Theory and Applications.P.Grindrod, OUP, 2015.