Distributed Immunisation Information System

Thesis Subtitle

198860

BSc Computer Science

School of Informatics and Engineering
Supervised by: Dr. Imran Khan
University of Sussex
2021

Abstract

..

Chapter 1

Statements

This report is submitted as part requirement for the degree of ... at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Chapter 2

Summary

Contents

List of Figures

Chapter 3

Introduction

(a) Motivation of the project
(b) Aims of the project, ideal place first paragraph
(c) Structure of the report

3.1 old intro

Inspired by the ideas of using blockchain to benefit society through decentralised applications in [5], this project seeks to design and develop a distributed solution to aid current immunisation information systems. As mentioned in [6], current implementations of IISs, especially that in developing countries, are lacking in terms of technological proficiency. As stated in [7], IISs are centralised repositories of personally identifiable vaccination information for individual members of a served population. This project aims to produce a decentralised version of an IIS, utilising Decentralised Ledger Technology (DLT) which has been highly discussed since the wide adoption of blockchain technology introduced by [1] - As discussed in [2]. From research such as [9] and [10], we know that blockchain provides the required security and privacy necessary for implementing these types of systems. [10] states that blockchain technology can reform the interoperability of healthcare databases. The system this project proposes uses a permissioned blockchain, in which the health authorities of different nations are trusted nodes in the network. Each health authority will provide immunisation records of their population to the ledger – such data will only be accessible by the health authority that provided it as well as the individual the record belongs to. This shall be enforced by using asymmetrical cryptography. A private key will be required to access these records, the key being held by the individual and their relevant health authority. This enables the individual's control over their immunisation records along with the ability to provide their private key and verify their immunisations. This, as discussed in [7], will simplify the processes of vaccination verification that may be required in pandemic scenarios,

registering for school, starting with a new employer, or crossing international borders. This system would benefit bodies striving to verify individuals' immunisations internationally. This report explores the required components of such a system, the professional considerations that are necessary for building this system such as legislation and ethical concerns, along with the requirements of an IIS and that of one implemented using blockchain.
Forgery and personal data security are dominant concerns of similar projects, but such problems are routinely solved for financial and other sensitive transactions. [3] This project attempts to utilise the techniques that achieve this, in other industries, to the vaccination passport problem.
As divulged in [2], DLT designs can be instantiated as a *public* or *private* distributed ledger [4], [5]. In public DLT designs, the underlying network allows arbitrary nodes to join and participate in the distributed ledger's maintenance. No registration or verification of the nodes' identities is required. Public DLT designs are often maintained by a large number of nodes, like with Bitcoin and Ethereum. The designs enable consistent high levels of availability. To allow for a high number of (arbitrary) nodes to find consensus, the designs for public DLT should be well scalable to not deter performance as more nodes join the network. [2]
In contrast, private DLT designs engage a defined set of nodes, with each node identifiable and known to the other network nodes. This means that private DLT designs require node verification when joining the distributed ledger, for example, by using Public Key Infrastructure (PKI). PKI comprises hardware, sfotware, policies, procedures and roles that are used for the secure electronic transfer of data by means of an insecfure network. A PKI manages the creation, distribution and revocation of digital certificates, which the use of public key cryptography requires. [2]
Blockchains can execute programmable transaction logic in the form of *smart contracts* as demonstrated by Ethereum [6]. The predacessor of smart contracts were the scripts in Bitcoin, introduced in [1]. Smart contracts function as *trusted distributed applications* and gains its security from teh blockchain and the underlying consensus among peers. This is similar to the approach of building resilient applications with state-machine replication (SMR) [7]. Though, blockchains are different from traditional SMR with Byzantine

3.2 Problem Space, blockchain intro

3.2.1 Blockchain architecture cryptography??

Randomized hashing offers the signer additional protection by reducing the likelihood that a preparer can generate two or more messages that ultimately yield the same hash value during the digital signature generation process – even if it is practical to find collisions for the hash function.
hash functions are formally defined in [8] as $afunctionh : D - - >$

$RwherethedomainD = 0, 1*, andtherangeR = 0, 1^n forsmoen >= 1$ using one way hash functions, as defined by Merkle in [9], are hash functions
store only hashes, this way data is kept anonymous and GDPR isn't a worry. Also, keeps the private data away from any verifiers etc because all they check against is a hash.
digital signatures $< --$ NEW!! NIST Publishes Special Publication 800-106 (Draft) "Randomized Hashing Digital Signatures". that documents our basic randomization technique for use with digital signatures. `http://csrc.nist.gov/publications/drafts/Draft-SP-800-106/Draft-SP800-106.pdf`

Chapter 4

Professional Considerations

(a) How did you address ethical issues?
(b) How did you address BCS Code of Conduct?
(c) If the interim report has mentioned that human participants would be involved but there was no need to apply for ethical approval, provide completed and signed form in appendix
The system proposed in this project will process personal data, in the form of immunisation records, so necessitates consideration of ethical and legal requirements.
Blockchains are not viable for data-storage, so ipfs (versinoed peer-to-peer file sharing system) or off-chain storage ?
`https://www.ibm.com/downloads/cas/RXOVXAPM` $<--$ talks about GDPR necessitating off-chain storage, because of Art. 17 GDPRRight to erasure ('right to be forgotten') `https://gdpr-info.eu/art-17-gdpr/` any research here? definitely necessitates incorporation of FHIR, or anything similar which is useful, to interoperability in terms of exististing Immunisation Information Systems infrastructure.

4.1 BCS Code of Conduct

This project has been aligned with the BCS Code of Conduct; relevant sections are as follows:

1. Public Interest You shall:

    a have a due regard for public health, privacy, security and wellbeing of others and the environment. Encryption methods will be used where necessary to ensure the confidentiality of information in the system.

    b have due regard for the legitimate rights of Third Parties. This system will make use of asymmetric-key encryption as well as hash functions to protect data, including that of third parties.

c promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise. This project proposes the system detailed be adopted by nations to provide a means of access to personal immunisation records, enabling ease of access for the population.

2. Professional Competence and Integrity You shall:

   a only undertake to do work or provide a service that is within your professional competence.

   b NOT claim any level of competence that you do not possess. This project has been thoroughly considered and the conclusion has been reached that it is within my professional competence.

   c develop your professional knowledge, skills and competence on a continuing basis. Maintaining awareness of technological developments, procedures, and standards that are relevant to your field. This project explores the current solutions for immunisation information systems, evolving my professional knowledge. This project displays an awareness of technological standards and procedures necessary for a distributed IIS.

   d ensure that you have the knowledge and understanding of Legislation and that you comply with such Legislation, in carrying out your professional responsibilities. This project includes a section exploring the legislation concerning this system. Specifically, the geographic area in which this system is being produced and how legislation governs the operations of such a system.

   e respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work. This project shall regularly be shared with my project supervisor to gain alternative viewpoints and criticisms, which will be implemented.

   f avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction. The design section of this project will detail the necessary security methods to maintain confidentiality, integrity and authenticity in the system.

4.1.1 Legislation

The General Data Protection Regulation (GDPR) is a privacy and security law, passed by the European Union (EU), imposes obligations onto organizations that target or collect data related to EU citizens and residents. [10]

The GDPR sets out several key principles:

1. Lawfulness, fairness and transparency

2. Purpose limitation
3. Data minimization
4. Accuracy
5. Storage limitation
6. Integrity and confidentiality
7. Accountability

These stated principles are essential to our approach to processing personal data. Compliance with the principles established in the GDPR is fundamental to ensuring good practice in data protection. In Article 35(1) of the GDPR it states that a Data Protection Impact Assessment (DPIA) is required "Where a type of processing in particular using new technologies, and taking into account the nature, scope, context and purposes of the processing, is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall, prior to the processing, carry out an assessment of the impact of the envisaged processing operations on the protection of personal data. A single assessment may address a set of similar processing operations that present similar high risks.". This asserts that if the system being designed and developed in this project is implemented in the real world performing a DPIA is mandatory. Though, a DPIA will not be necessary for this undertaking. As the GDPR is mainly concerned with the European Economic Area (EEA), producing this system in the UK brings concerns. The UK is currently in a transition period until the 31st of December 2020. At the end of this transition period the UK will become a third country. Presently, the UK is seeking adequacy decisions from the European Commission. "The effect of an adequacy decision is that personal data can be sent from an EEA state to a third country without any further safeguard being necessary" because "The European Commission has the power to determine whether a third country has an adequate level of data protection." [3]. If the adequacy decision is not secured, by the end of the transition period, the provisions set out in [4] will take effect.
Medical records are pieces of personal, sensitive data which are stored, processed, processed and transmitted in healthcare systems.
Recital 26 Not Applicable to Anonymous Data* so basically don't worry about GDPR `https://gdpr-info.eu/recitals/no-26/`

4.1.2 How did you address ethical issues?

4.1.3 How did you address BCS Code of Conduct?

4.1.4 Ethical Implications

"Issues of concern are falsified or counterfeit vaccine certificates" are described in [11], when considering digital vaccination passports. Blockchain enables an

immutable store of data, rendering any attempt at providing a spoofed record would be nullified due to the data the individual provides not being in the ledger. Hyperledger Fabric, the framework of choice to aid in production of the network, allows access to verify that private data is in fact in a ledger without revealing it.

4.1.5 Data Privacy

other than GDPR, considerations include: what?

Chapter 5

Body of Report

5.1 blockchian stuff

[12] introduced distributed networks, using a new concept in which computers are connected to other stations rather than switching points like in a centralised system.
Distributed systems are defined as a system in which hardhware or software components located at networked computers communicatte and coordinate their actions via message passing. [13]
As stated in [13], a distributed system must accommodate heterogeneous hardware, operating systems and networks. The networks may differ widely in performance. Systems of widely differing scales, ranging from tens of computers to millions of computers, must be supported.
Replication is key to the effectiveness of distributed systems, it can provide enhanced performance, fault tolerance and high availability. [13]
nodes in distributed systems can be honest, faulty or mailcious. [14] [13] defines arbitrary failures as follows. The term arbitrary or Byzantine failure is used to describe the worst possible failure semantics, in which any type of error may occur. For example, a process may set wrong values in its data items, or it may return a wrong value in response to an invocation.
In [15] the thought experiment of The Byzantine Generals Problem was proposed. The scenerio depicts a group of army generals, leading different parts of the Byzantine army plan to attack or retreat from a city. The generals can only communicate with eachother via messenger. Communication is necessary in agreeing on a strategy to avoid failure. The problem being that one or more of the generals may be traitorous and is attempting to prevent the others from reaching an agreement. This necessitates a viable mechanism that enables agreement between actors in the presence of traitors. Analogously, distributed systems require agreement amongst nodes, whilst using a channel for communication, even in the presence of Byzantine nodes. This is known as the consensus problem. [16] The consensus problem occurs when attempting

to achieve reliability in a distributed system, in the presence of faulty processes. A system requires processes to reach an *a*greement on a value after one or more processes propose what the value should be. In a system such that: each process p_i communicates with other processes via message passing (assuming communication is reliable), up to some number f of the N processes are faulty, the remainder of processes are correct.
Reaching consensus is achieved as follows. Every process p_i starts in an *undecided* state and *proposes* a single value v_i. The processes communicate with eachother and exchange values. Each process then sets the value of a *decision variable*, d_i. In doing so the process enters the *decided* state and can no longer change d_i.[13]

1. Termination: Eventually each correct process sets a decision variable.
2. Agreement: The decision of all correct processes is the same.
3. Integrity: If the correct processes al proposed the same value, then any correct process in the *decided* state has chosen that value.[13]

The Byzantine Generals Problem was solved in [17], where the Practical Byzantine Fault Tolerance (PBFT) algorithm was introduced.
The first practical implementation of PBFT was produced with the inception of Bitcoin, in which the Proof-of-Work (PoW) algorithm was developed as a consensus mechanism. [1]
In [18], Raft was introduced as a consensus algorithm, which produced similar results to and was as efficient as Paxos - a family of protocols for solving consensus, presented in [19]. Raft was a successful attempt at restructuring a consensus mechanism in order to enhance understandability. Raft is crash fault tolerant, but lacks in Byzantine fault tolerance. [18]
Ledger is the sequenced, tamper-proof ...
chain is transaction log, structured as hash-linked blocks where each block contains a sequence of N transactions.
Rethinking Permissioned Blockchains [20]

5.1.1 Healthcare Stuff

use these:
[21] $<--$ Implementation Considerations for Blockchain in Healthcare Institutions
[22] $<--$ 'Comparison of Smart Contract Blockchains for Healthcare Applications'
[23] $<--$
'A-Privacy-Preserving-Healthcare-Framework-Using-Hyperledger-Fabric'

FHIR

FHIR - standards for sharing healthcare information.

5.1.2 Security

a reference = [24] < −− 'On the Security and Privacy of Hyperledger Fabric - Challenges and Open Issues'
Whilst penetration testing Hyperledger Fabric, there was found a high severity issue, which describes the potential to guess the content of a Private Data Collection. This is achieved using SHA256 as an oracle. To ensure this vulnerability is not available in the system, it is necessary to ensure any private data is salted before being hashed. [25]

5.2 benefits

[Covid-19: Vaccines and vaccine passports being sold on darknet] - this system would negate these spoofed vaccination records as all data will be on the system and if not it's illegitimate.

5.3 System Design

Usage of Raft or Kafka is sufficient as Byzantine fault tolerance is not required in the proposed sysetem, as the nodes shall be controlled by trusted authorities. Though, a Byzantine fault-tolerant ordering service for Hyperledger Fabric has been developed, this would be useful if nodes were prone to malicious activity. [26]
[27] states that Raft is superior to Kafka in terms of success and speed when conducting invoke transactions due to its simpller framework. But, when querying transactions Kafka is superior since the throughput rate is larger. So, which one ??
Developing and testing chaincode was initially performed using the Fabric test network, provided in the official documentation for Hyperledger Fabric [28], contained in the "fabric-samples" repository A.1. It is based on a limited configuration:

1. It includes two peer organizations and an ordering organization.
2. For simplicity, a single node Raft ordering service is configured.
3. To reduce complexity, a TLS Certificate Authority (CA) is not deployed. All certificates are issued by the root CAs.
4. The sample network deploys a Fabric network with Docker Compose. Because the nodes are isolated within a Docker Compose network, the test network is not configured to connect to other running Fabric nodes. [28]

Currently, medical data has the HL7 standard as a data exchange format but the exchange of medical records between countries is not widespread. [29]

5.3.1 Default application SDK

Go is best for chaincode but what would be the most suitable for the access applications? Java, because it's already highly used in enteprise already? **find reference for this ** Java also enables cross-platform usage, most convenient for these systems that are already up and running. - Node satisfies these conditions also, Node has Performance Traffic Engine in fabric-test toolset. Fabric Network Operator tool, from Hyperledger, also utillises the Node SDK, looking like the project might just wanna use Node? Benefits to web app vs desktop? Java is probs more secure ** find ref ** but web app with Node would likely be more flexible, as it can be used for the access application as any device runs Node but internet is required. Java is statically typed, but this is also achieved when using TypeScript with Node - this brings both SDKs to an equal playing field, in terms of maintainability. Orgs will be able to develop their own applications, enabled by OSS. However, considerations are in place for the best default SDK to use, as the correct choice could simplify adoption.
Does Node possess an advantage in terms of ease of adoption? Can more mobile devices use Node than Java? Does it even matter which SDK?

5.3.2 Hyperledger Fabric

Hyperledger fabric is ... Using their MSPs is different to things like this `'https://www.hyperledger.org/blog/2020/04/21/trustid-a-new-approach-to-fabric-user-identity-management'` TrustID: A New Approach to Fabric User Identity Management how? Why are we not customising and instead going with fabric's defaut implementation, maybe because it fits our use case better? It makes sense as the orginisations using the system are governed already by their heath body, for UK it's MAYBE Department of Health & Social Care - look this up! See if there's a general term for lead health body or whatever
Fabric has all the identity functionality built in, Indy + Aries would be more flexible but does that matter with an permissioned network? Fabric MSPs etc vs DIDs? Or does it use DIDs? `'https://www.w3.org/TR/did-core/'` w3 DIDs
Transaction flow "The SDK serves as a shim to package the transaction proposal into the properly architected format (protocol buffer over gRPC) and takes the user's cryptographic credentials to produce a unique signature for this transaction proposal." - `"https://hyperledger-fabric.readthedocs.io/en/latest/txflow.html'`.

Enforcing Determinism of Java Smart Contracts [30]
`'https://wiki.hyperledger.org/display/fabric/Design+Documents'`
WRITE ABOUT THIS!
Ensuring determinism in smart contracts is essential
This first phase also eliminates any non-determinism, as inconsistent results

can be filtered out before ordering. Because we have eliminated non-determinism, Fabric is the first blockchain technology that enables use of standard programming languages. [31]

Architecture Overview

Hyperledger Fabric's key design features, as introduced in [31], are explored in the following text. Asset, defined by chaincode, are exchanged across the network. These can range frmo the tangible to teh intangible. These are stored as a collection of key-value pairs, represented in binary and/or JSON, with state changes recorded as transaction on a channel ledger. Chaincode, the software defining assets, also defines the transaction instructions for manipulating the assets; chaincode is the business logic. Invocation of chaincode is recorded in the ledger. The ledger is described in [32] as the "sequenced, tamper-resistant record of all state transitions in the fabric." This ledger is comprised of a blockchian to store the records in blocks and a state database. Channels are used to communicate privately between organisations, enabling privacy. Each channel contains a ledger, for that channel, peers also maintain a copy of the ledger for each channel of which they participate. This is used to maintain consistency across peers.
Hyperledger projects follow a design philosophy, which includes a modular, extinsible approach; enabling interoperability. Design puts an emphasis on secure solutions, crucial to systems using sensitive data. Hyperledger's token-agnostic approach simplifies bringing blockchain to business infrastructure. [33]
Hyperledger projects embrace security by design and follow the best practices specified by the Linux Foundation's Core Infrastructure initiative.
`https://www.coreinfrastructure.org/`
As discussed in [31], blockchain systems typically, both permissioned and permissionless, follow the order-execute architecture. This execution style involves ordering transactions first, using a consensus protocol, then executes them in the same order on all peers sequentially.[1]

[1]This report follows the convention set in the Hyperledger Fabric whitepaper of calling the transaction execution (named "transaction validation" in blockchains such as Bitcoin) *transaction execution* to bring the terminology together.

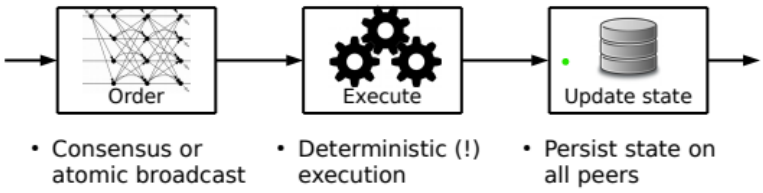

Figure 5.1: Order-execute architecture in replicated services

Source: [31]

Whilst the order-execute architecture is conceptually simple and therefore widely used, there are drawbacks which occur when using it for a general-purpose permissioned blockchain.[31] The most significant disadvantages are as follows: *Sequential execution.* Executing transactions sequentially on all peers limits the effective throughput that can be achieved on the blockchain.[31] One solution to this problem, used by Ethereum [34], utilises the concept of *gas* consumed by a transaction execution, which is converted at a *gas price* to a cost in the cryptocurrency and billed to the transaction submitter. This solution is excellent for use in permissionless blockchains, though does not suffice when designing a general-purpose, permissioned, token-agnostic, blockchain like Hyperledger Fabric. [31]

Hyperledger Fabric distributed applications consist of two parts: A smart contract, called *chaincode*, which is program code that implements the application logic and runs during the *execution phase*. As well as an *endorsment policy* that is evaluated in the *validation phase*. [31]

This system is utilising v2.2, the LTS release of the second major release of Fabric. Fabric v2.0 delivers important new features and changes for users and operators. Namely, the addition of decentralised governance for smart contracts. A new implementation of the chaincode lifecycle allows multiple organisations to agree on parameters of chaincode, before it can be used to interact with the ledger. These parameters could be ... [35]
In Fabric, to participate, every node and user that interacts with a network needs to be part of an organisation. [28]
Members of a network in Fabric enroll through a trusted Membership Service Provider (MSP). [36] An MSP serves as a trusted authority, that is in control of the governing of identities for its organisation.
An MSP is a structure of folders added to the network configuration, to define the permissions and roles in an organisation. Certificate Authorities generate the certificates that represent identities, the MSP is where these reside. Roles in organisations are assigned to the identities, by the MSP. MSPs also hold a copy of the revoked certificates, for the organisation, which is checked whenever the certificate is attempting to be used. [37] This list is called a

Certificate Revocation List (CRL) [38] and is stored by the issuing CA. [39] MSPs come in two flavors, *local MSPs* have a scope limited to the organisation they are part of and reside on the organisation's node. Every node requires an MSP. *Channel MSPs* define administrative and participatory rights at a channel level. These channel MSPs include the MSPs of the organisations on the channel, enabling the control of relationships between the channel participants. [37]
Fabric CA is the default Certificate Authority in a Fabric network. Acting as a root CA, Fabric CA issues the certificatse required for the MSPs to implementend identities and roles. [39]

Privacy by Design

[40] $<--$ private data collections, is privacy by design - enables GDPR compliancy etc = [41] $<--$ Privacy by Design Helps Blockchains Comply With GDPR
Hyperledger Fabric has an excellent offering of privacy protection features. Asymmetric cryptography and zero-knowledge proof separate the transaction data from records on chain. Identities for network actors are encapsulated in X.509 digital certificates and are controled by MSPs as aforementioned. [39]. A X.509 digital certificate or public key certificate is a signed document which holds information on the identity to which the certificate has been issued, as well as the identity that issued it known as a Certificate Authority (CA). These certificates are based on the widely accepted X.509 Public Key Infrastructure (PKI). [38] Fabric utilises certificate chains, a concept defined using the name *certification path* in [38], which Hyperledger calls a *Chain of trust*, to distribute the responsibiity of issuing certificates. The processes uses *Intermediate CAs* which have their certificates issued by a Root CA. This allows for backtracking to find the Root CA of any issued certificate, this helps scaling remain secure and limits the exposure of Root CAs - which if compromised would endanger the *Chain of trust*. [39] The *Chain of trust* is depicted in 5.2.

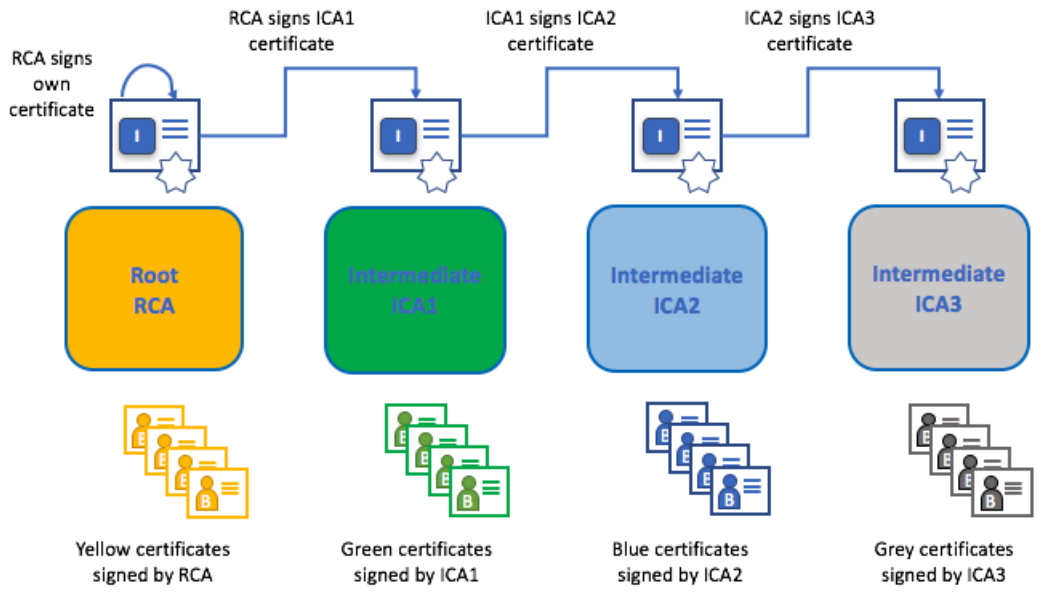

Figure 5.2: CA chain of trust in Fabric

Source: [39]

[42] <−− The privacy protection mechanism of Hyperledger Fabric and its application in supply chain finance

Inspiration

Use −− > [43]. ¡– says there is a "LACK OF UNIVERSALLY DEFINED STANDARDS" is this true still? should this report be proposing standards for blockchain healthcare systems, are there thusfar?

Consensus

In Hyperledger business blockchain frameworks consensus is reached by performing two seperate activities:

1. Ordering of transactions
2. Validating transactions

In the first step, the transactions are received from the client. An ordering service is used to order the transactions. To enable confidentiality, the ordering service may be agnostic to the transaction; that is, the transaction content can be hashed or encrypted.[33] This is extremely benificial to this system, as maintaining confidentiality is essential to abiding by the GDPR [10].

Consensus in Hyperledger Fabric is further broken into 3 phases: *Endorsement*, *Ordering* and *Validation*.

1. Endorsement is driven by policy (eg. m of n signatures) upon which participants endorse a transaction.

2. Ordering phase accepts the endorsed transactions and agrees to the order to be committed to teh ledger.

3. Validation takes a block of ordered transactions and validates the correctness of the results, including checking endorsement policy and double-spending.

zero-knowledge asset transfer vs zero-knowledge proof?
Using Go chaincode, generating UML diagrams from this, classes such as: Record, with states: immunised or not?

5.4 System Architecture Overview

going to call the "admissions staff" (people who want to check immunisation information) Verifiers.
Each organisation in the network has control over its members, through Fabric's MSPs. This control also extends to the organisation's data, this shall not be shared with other organisations unless allowed by the organisation. Private channel capabilities ensure that data will only be shared with those authorised by the owning organisation.
Fabric's test-network was used in the initial design of the network for the DIIS A.1. The test network, as described previously, enables the testing of chaincode and applications without setting up a dedicated network. This is excellent for the development process.

5.5

5.6 Chaincode

[44] suggests that medical records should contain metadata of a patient-provider encounter (visit date/time, location, etc.), the data stored off-chain and should be entered when the record is produced. As immunisations expire, the ledger must store metadata on immunisations to ensure the record in the ledger can be invalidated when the date is reached.
Chaincode is Hyperlegder Fabric's version of smart contracts. Chaincode is a program, which can be written in general-programming languages Go, JavaScript (for Nodejs runtime) or Java, that implements a prescribed interface.
Selected Go, as there are apparent benifits outlined in [45]

The duration of protection conferred by vaccines should be tied to passport expiry dates [3]. This is implemented via the expiration of our Record contract.
The chaincode for this system will represent "Records", immunisation records, the initial chaincode package will be called "record_1.0". The implementation of this is documented below. Representing the records as chaincode, or smart contracts, enables the manipulation of records by organisations that "own" them. The "Owner" shall be recorded inside the Record contract. The chaincode written for this system can be found at A.2.

5.6.1 Implementation

Fabric networks use chaincode to initialise and manage the ledger state, through transactions submitted by applications. [46] To use chaincode in a network organisations need to agree to the parameters, the name of the chaincode, version and the endorsement policy. [46] explains that the agreement is reached using the following steps:

1. Package the Chaincode
2. Install the chaincode on peers: Each organisation that will utilise the specific chaincode, to endorse transactions or query ledgers, needs to execute this step.
3. Approve chaincode definition for organisation: This step also requires completion from each organisation that will use the chaincode. By default, the chaincode needs to be approved by atleast a majority of organisations before it can be deployed on a channel.
4. Commit the chaincode definition to the channel: After the chaincode is approved by a sufficient number of organisations, one organisation can commit the chaincode definition.

Chaincode needs to be packaged in a tar file for it to be installed onto peers [47]. Using the Fabric peer binaries, the following command can be used to skip manually adding the necessary "metadata.json" file which required in the tar file.

Listing 5.1: peer lifecycle chaincode package command

```
# create the chaincode package using the peer
    lifecycle chaincode package command
peer lifecycle chaincode package record.tar.gz --
    path ../asset-transfer-basic/chaincode-go/ --lang
     golang --label record_1.0
```

Though not necessary for Fabric networks, it would be useful for organisations to synchronise their labels for chaincode. This will be a standard for adopting this system.

Installation of chaincode on peers can be executed via CLI or an SDK [47].

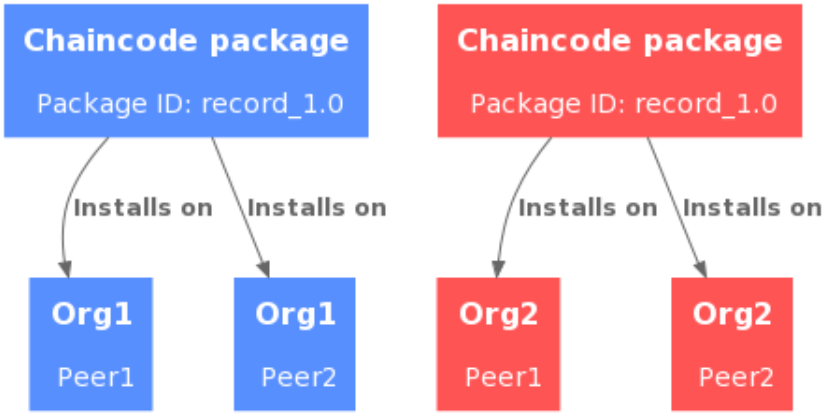

Figure 5.3: Peer admin from two organisations installs the chaincode package record_1.0

Source: Adapted from [47]

In this system is it not necessary to set the endorsement policy for the chaincode as it will be utlising the default value, which requires that a majority of organisations endorse a transaction. [47]

5.7 Immunisation Verification

smarthealth.cards by Vaccination Credential Initiative (VCI) - smart card framework.

Use an SDK to invoke chaincode that compares the provided data (hash of the transaction? - in which your immunisation record is stored) with the hash (SHA-250 probs) of the data in the ledger. If there's no match there's no verification, details?

5.8 Blockchian-as-a-Service

Using Blockchain for Electronic Health Records `'https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8863359'` [48] **shortcomings are discussed here: `'https://ieeexplore.ieee.org/document/9284197'` [24] ** . The computing power and security provided by BaaS can make up for the shortcomings aforementioned.[49].

Comparing Blockchain-as-a-Service platforms, to identify which tool is most suitable for delpoying the proposed Distributed Immunisation Information System built with Hyperledger Fabric.

Reading [50].

[Performance Analytical Comparison of Blockchain-as-a-Service (BaaS) Platforms] says "Blockchain highly suffers from scalability problem due to its capped transaction latency as well as consensus approach"

Azure have FHIR API, is it the only one? looks like maybe Amazon also `'https://azure.microsoft.com/en-gb/services/azure-api-for-fhir/?ocid=AID754288&wt.mc_id=azfr-c9-scottha%2CCFID0475'`Azure FHIR API

Beginning, I was aware of two options: Azure and IBM.

5.8.1 IBM Blockchain Platform

To ensure all organisations can setup the necessary infrastructure, it's most simple to use a Blockchain-as-a-Service platform. IBM Blockchain Platform for IBM Cloud is built with Hyperledger Fabric v1.4.11 and v2.2.2. The system proposed in this report uses v2.2.2, as it delivers delivers important new features and changes for users and operators alike, including support for new application and privacy patterns, enhanced governance around smart contracts, and new options for operating nodes. [35]
useful for tesing, this uses This code pattern uses IBM Db2 federation capabilities to perform SQL analytics on a sample blockchain insurance application using Hyperledger Fabric. This code pattern uses the sample blockchain insurance application. Apache Zeppelin notebook will be used to perform analytics by querying the blockchain using Db2 and SQL. - https://developer.ibm.com/patterns/use-db2-and-sql-to-perform-analytics-on-blockchain-transactions/
May have to use IBM just because of their whole hybrid cloud infrastructure, could be too easy to give up [51]
Requested demo of IBM Health Pass, will look more into this physically when the change occurs. For now, review the literature provided publicly - not much lmao `'https://www.ibm.com/watson/health/resources/digital-health-pass-blockchain-explained/' 'https://mytechdecisions.com/compliance/ibm-salesforce-work-covid-19/'`

Chapter 6

Conclusion

This should be an assessment of the finished product.
Have you met your objectives? If not, why not?
Suggestions for future extensions
Alternatives methodologies might lead to a better system

Bibliography

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Nov. 20, 2019, Publication Title: Manubot. [Online]. Available: https://git.dhimmel.com/bitcoin-whitepaper/ (visited on 11/18/2020).

[2] A. Sunyaev, "Distributed ledger technology," in *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, Cham: Springer International Publishing, 2020, pp. 265–299, ISBN: 978-3-030-34957-8. DOI: 10.1007/978-3-030-34957-8_9. [Online]. Available: https://doi.org/10.1007/978-3-030-34957-8_9.

[3] C. Dye and M. C. Mills, "COVID-19 vaccination passports," *Science*, vol. 371, no. 6535, pp. 1184–1184, Mar. 19, 2021, Publisher: American Association for the Advancement of Science, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abi5245. [Online]. Available: https://science.sciencemag.org/content/371/6535/1184 (visited on 04/23/2021).

[4] K. Yeow, A. Gani, R. W. Ahmad, J. J. P. C. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2018, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2779263.

[5] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2017, pp. 243–252. DOI: 10.1109/ICSA.2017.33.

[6] (). Ethereum whitepaper, ethereum.org, [Online]. Available: https://ethereum.org (visited on 04/22/2021).

[7] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, Dec. 1, 1990, ISSN: 0360-0300. DOI: 10.1145/98163.98167. [Online]. Available: https://doi.org/10.1145/98163.98167 (visited on 04/22/2021).

[8] B. V. Rompay and K. U. L. Esat, "Analysis and Design of Cryptographic Hash Functions, MAC Algorithms and Block Ciphers," p. 259,

[9] R. C. Merkle, "Secrecy, authentication, and public key systems.," AAI8001972, PhD thesis, Stanford University, Stanford, CA, USA, 1979, 187 pp.

[10] (). General data protection regulation (GDPR) – official legal text, General Data Protection Regulation (GDPR), [Online]. Available: https://gdpr-info.eu/ (visited on 04/04/2021).

[11] P. Schlagenhauf, D. Patel, A. J. Rodriguez-Morales, P. Gautret, M. P. Grobusch, and K. Leder, "Variants, vaccines and vaccination passports: Challenges and chances for travel medicine in 2021," *Travel Medicine and Infectious Disease*, vol. 40, p. 101 996, 2021, ISSN: 1477-8939. DOI: 10.1016/j.tmaid.2021.101996. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7899929/ (visited on 04/03/2021).

[12] P. Baran, "On distributed communications networks," *IEEE Transactions on Communications Systems*, vol. 12, no. 1, pp. 1–9, Mar. 1964, Conference Name: IEEE Transactions on Communications Systems, ISSN: 1558-2647. DOI: 10.1109/TCOM.1964.1088883.

[13] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*. Pearson Education, Nov. 21, 2011, 1066 pp., Google-Books-ID: 3ZouAAAAQBAJ, ISBN: 978-0-13-300137-2.

[14] I. Bashir, *Mastering Blockchain*. Packt Publishing Ltd, Mar. 17, 2017, 531 pp., Google-Books-ID: urkrDwAAQBAJ, ISBN: 978-1-78712-929-0.

[15] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, Jul. 1, 1982, ISSN: 0164-0925. DOI: 10.1145/357172.357176. [Online]. Available: https://doi.org/10.1145/357172.357176 (visited on 04/10/2021).

[16] M. J. Fischer, "The consensus problem in unreliable distributed systems (a brief survey)," in *Foundations of Computation Theory*, M. Karpinski, Ed., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1983, pp. 127–140, ISBN: 978-3-540-38682-7. DOI: 10.1007/3-540-12689-9_99.

[17] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the third symposium on Operating systems design and implementation*, ser. OSDI '99, USA: USENIX Association, Feb. 22, 1999, pp. 173–186, ISBN: 978-1-880446-39-3. (visited on 04/24/2021).

[18] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14, USA: USENIX Association, Jun. 19, 2014, pp. 305–320, ISBN: 978-1-931971-10-2. (visited on 04/24/2021).

[19] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1, 1980, ISSN: 0004-5411. DOI: 10.1145/322186.322188. [Online]. Available: https://doi.org/10.1145/322186.322188 (visited on 04/24/2021).

[20] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, ser. BCC '17, New York, NY, USA: Association for Computing Machinery, Apr. 2, 2017, pp. 3–7, ISBN: 978-1-4503-4974-1. DOI: 10.1145/3055518.3055526. [Online]. Available: https://doi.org/10.1145/3055518.3055526 (visited on 04/24/2021).

[21] K. Paranjape, M. Parker, D. Houlding, and J. Car, "Implementation considerations for blockchain in healthcare institutions," *Blockchain in Healthcare Today*, Jul. 9, 2019, ISSN: 2573-8240. DOI: 10.30953/bhty.v2.114. [Online]. Available: https://blockchainhealthcaretoday.com/index.php/journal/article/view/114 (visited on 04/18/2021).

[22] H. Yu, H. Sun, D. Wu, and T.-T. Kuo, "Comparison of smart contract blockchains for healthcare applications," *AMIA Annual Symposium Proceedings*, vol. 2019, pp. 1266–1275, Mar. 4, 2020, ISSN: 1942-597X. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7153130/ (visited on 04/03/2021).

[23] C. Stamatellis, P. Papadopoulos, N. Pitropakis, S. Katsikas, and W. J. Buchanan, "A privacy-preserving healthcare framework using hyperledger fabric," *Sensors*, vol. 20, no. 22, p. 6587, Jan. 2020, Number: 22 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/s20226587. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6587 (visited on 04/10/2021).

[24] S. Brotsis, N. Kolokotronis, K. Limniotis, G. Bendiab, and S. Shiaeles, "On the security and privacy of hyperledger fabric: Challenges and open issues," in *2020 IEEE World Congress on Services (SERVICES)*, ISSN: 2642-939X, Oct. 2020, pp. 197–204. DOI: 10.1109/SERVICES48979.2020.00049.

[25] G. Shaw, "Penetration testing technical report," p. 20, Aug. 8, 2019.

[26] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, ISSN: 2158-3927, Jun. 2018, pp. 51–58. DOI: 10.1109/DSN.2018.00018.

[27] H. Yusuf1 and I. Surjandari, "Comparison of performance between kafka and raft as ordering service nodes implementation in hyperledger fabric," *International Journal of Advanced Science and Technology*, vol. 29, no. 7, pp. 3549–3554, May 1, 2020, Number: 7s, ISSN: 2005-4238. [Online]. Available: http://sersc.org/journals/index.php/IJAST/article/view/17652 (visited on 04/24/2021).

[28] (). Using the fabric test network — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html (visited on 04/23/2021).

[29] H. H. Kung, Y.-F. Cheng, H.-A. Lee, and C.-Y. Hsu, "Personal health record in FHIR format based on blockchain architecture," in *Frontier Computing*, J. C. Hung, N. Y. Yen, and J.-W. Chang, Eds., ser. Lecture Notes in Electrical Engineering, Singapore: Springer, 2020, pp. 1776–1788, ISBN: 9789811532504. DOI: 10.1007/978-981-15-3250-4_237.

[30] F. Spoto, "Enforcing determinism of java smart contracts," in *Financial Cryptography and Data Security*, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 568–583, ISBN: 978-3-030-54455-3. DOI: 10.1007/978-3-030-54455-3_40.

[31] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, Apr. 23, 2018. DOI: 10.1145/3190508.3190538. arXiv: 1801.10228. [Online]. Available: http://arxiv.org/abs/1801.10228 (visited on 11/19/2020).

[32] (). Hyperledger fabric model — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric_model.html (visited on 04/25/2021).

[33] "Hyperledger architecture, volume 1," Aug. 2017. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf (visited on 04/11/2021).

[34] D. G. Wood, "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER," p. 39,

[35] (). What's new in hyperledger fabric v2.x — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatsnew.html (visited on 04/22/2021).

[36] (). Introduction — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/blockchain.html (visited on 04/25/2021).

[37] (). Membership service provider (MSP) — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/membership/membership.html (visited on 04/25/2021).

[38] D. Cooper. (). Internet x.509 public key infrastructure certificate and certificate revocation list (CRL) profile, [Online]. Available: https://tools.ietf.org/html/rfc5280 (visited on 04/24/2021).

[39] (). Identity — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/identity/identity.html (visited on 04/24/2021).

[40] (). Private data — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html (visited on 04/24/2021).

[41] H. Narumanchi and N. Emmadi, "Privacy by design helps blockchains comply with GDPR," Dec. 28, 2019.

[42] C. Ma, X. Kong, Q. Lan, and Z. Zhongding, "The privacy protection mechanism of hyperledger fabric and its application in supply chain finance," *Cybersecurity*, vol. 2, Dec. 1, 2019. DOI: 10.1186/s42400-019-0022-2.

[43] P. Yuan, X. Xiong, L. Lei, and K. Zheng, "Design and implementation on hyperledger-based emission trading system," *IEEE Access*, vol. PP, pp. 1–1, Dec. 20, 2018. DOI: 10.1109/ACCESS.2018.2888929.

[44] S. Alexaki, G. Alexandris, V. Katos, and N. E. Petroulakis, "Blockchain-based electronic patient records for regulated circular healthcare jurisdictions," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, ISSN: 2378-4873, Sep. 2018, pp. 1–6. DOI: 10.1109/CAMAD.2018.8514954.

[45] L. Foschini, A. Gavagna, G. Martuscelli, and R. Montanari, "Hyperledger fabric blockchain: Chaincode performance analysis," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, ISSN: 1938-1883, Jun. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149080.

[46] (). Writing your first chaincode — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode4ade.html (visited on 04/26/2021).

[47] (). Fabric chaincode lifecycle — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode_lifecycle.html (visited on 04/26/2021).

[48] A. Shahnaz, U. Qamar, and A. Khalid, "Using blockchain for electronic health records," *IEEE Access*, vol. 7, pp. 147 782–147 795, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2946373.

[49] J. Song, P. Zhang, M. Alkubati, Y. Bao, and G. Yu, "Research advances on blockchain-as-a-service: Architectures, applications and challenges," *Digital Communications and Networks*, Feb. 20, 2021, ISSN: 2352-8648. DOI: 10.1016/j.dcan.2021.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864821000092 (visited on 04/04/2021).

[50] M. M. H. Onik and M. H. Miraz, "Performance analytical comparison of blockchain-as-a-service (BaaS) platforms," in *Emerging Technologies in Computing*, M. H. Miraz, P. S. Excell, A. Ware, S. Soomro, and M. Ali, Eds., ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Cham: Springer International Publishing, 2019, pp. 3–18, ISBN: 978-3-030-23943-5. DOI: 10.1007/978-3-030-23943-5_1.

[51] "IBM blockchain platform technical overview," May 2020. [Online]. Available: https://www.ibm.com/downloads/cas/Q9DGBLV7 (visited on 04/12/2021).

Appendix A

Hyperledger Fabric

A.1 Test network

Used for initial testing of chaincode, available on GitHub [1]

A.2 Chaincode

The chaincode developed for the DIIS, representing immunisation records. [2]

Listing A.1: Chaincode representing immunisation records.

```
package main

import (
  "encoding/json"
  "fmt"
  "log"
  //"time"

  "github.com/hyperledger/fabric-contract-api-go/
     contractapi"
)

// Record provides functions for managing an Asset
   type Record struct {
      contractapi.Contract
    }
```

[1] https://github.com/hyperledger/fabric-samples/tree/main/test-network

[2] Adapted from tutorial provided in official documentation for Hyperledger Fabric "https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode4ade.html"

```
// medical records should contain metadata of a
   patient-provider encounter (visit date/time,
   location, etc.),

// Asset describes basic details of what makes up a
   simple asset
   type Asset struct {
      ID             string `json:"ID"`
      Timestamp          string `json:"dateTime"`
      Owner          string `json:"owner"`
      Expiration     uint64    `json:"expiration"`
    }

// InitLedger adds a base set of assets to the ledger
   func (s *Record) InitLedger(ctx contractapi.
      TransactionContextInterface) error {
    assets := []Asset{
      {ID: "asset1", Timestamp: "2012-10-31
         15:50:13.793654 +0000 UTC", Owner: "Tomoko",
         Expiration: 104467440737095516},
      {ID: "asset2", Timestamp: "2011-10-21
         12:07:13.793654 +0000 UTC", Owner: "Brad",
         Expiration: 438579485745748},
      {ID: "asset3", Timestamp: "2009-01-11
         13:12:13.263674 +0000 UTC", Owner: "Jin Soo",
          Expiration: 89273489237483},
      {ID: "asset4", Timestamp: "2020-03-13
         11:03:13.795664 +0000 UTC", Owner: "Max",
         Expiration: 239048203894},
      {ID: "asset5", Timestamp: "2019-11-05
         14:12:11.798454 +0000 UTC", Owner: "Adriana",
          Expiration: 9023423948333},
      {ID: "asset6", Timestamp: "2013-05-21
         10:01:13.274683 +0000 UTC", Owner: "Michel",
         Expiration: 290378489237},
    }

    for _, asset := range assets {
      assetJSON, err := json.Marshal(asset)
      if err != nil {
        return err
      }

      err = ctx.GetStub().PutState(asset.ID, assetJSON
         )
      if err != nil {
```

```
        return fmt.Errorf("failed to put to world
            state. %v", err)
      }
    }

    return nil
  }

// CreateAsset issues a new asset to the world state
   with given details.
   func (s *Record) CreateAsset(ctx contractapi.
      TransactionContextInterface, id string,
      timestamp string, owner string, expiration
      uint64) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
      return err
    }
    if exists {
      return fmt.Errorf("the asset %s already exists",
          id)
    }

    asset := Asset{
      ID: id,
      Timestamp:  timestamp,
      Owner:  owner,
      Expiration: expiration,
    }
    assetJSON, err := json.Marshal(asset)
    if err != nil {
      return err
    }

    return ctx.GetStub().PutState(id, assetJSON)
  }

// ReadAsset returns the asset stored in the world
   state with given id.
   func (s *Record) ReadAsset(ctx contractapi.
      TransactionContextInterface, id string) (*Asset,
       error) {
    assetJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
      return nil, fmt.Errorf("failed to read from
         world state: %v", err)
```

```
    }
    if assetJSON == nil {
      return nil, fmt.Errorf("the asset %s does not
          exist", id)
    }

    var asset Asset
    err = json.Unmarshal(assetJSON, &asset)
    if err != nil {
      return nil, err
    }

    return &asset, nil
  }

// UpdateAsset updates an existing asset in the world
   state with provided parameters.
   func (s *Record) UpdateAsset(ctx contractapi.
      TransactionContextInterface, id string,
      timestamp string, owner string, expiration
      uint64) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
      return err
    }
    if !exists {
      return fmt.Errorf("the asset %s does not exist",
          id)
    }

    // overwriting original asset with new asset
    asset := Asset{
      ID: id,
      Timestamp:  timestamp,
      Owner:  owner,
      Expiration: expiration,
    }
    assetJSON, err := json.Marshal(asset)
    if err != nil {
      return err
    }

    return ctx.GetStub().PutState(id, assetJSON)
  }
```

```
  // DeleteAsset deletes an given asset from the world
      state.
  func (s *Record) DeleteAsset(ctx contractapi.
     TransactionContextInterface, id string) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
      return err
    }
    if !exists {
      return fmt.Errorf("the asset %s does not exist",
          id)
    }

    return ctx.GetStub().DelState(id)
  }

// AssetExists returns true when asset with given ID
    exists in world state
   func (s *Record) AssetExists(ctx contractapi.
      TransactionContextInterface, id string) (bool,
      error) {
    assetJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
      return false, fmt.Errorf("failed to read from
         world state: %v", err)
    }

    return assetJSON != nil, nil
  }

// TransferAsset updates the owner field of asset with
     given id in world state.
   func (s *Record) TransferAsset(ctx contractapi.
      TransactionContextInterface, id string, newOwner
       string) error {
    asset, err := s.ReadAsset(ctx, id)
    if err != nil {
      return err
    }

    asset.Owner = newOwner
    assetJSON, err := json.Marshal(asset)
    if err != nil {
      return err
    }
```

```
    return ctx.GetStub().PutState(id, assetJSON)
  }

// GetAllAssets returns all assets found in world
   state
   func (s *Record) GetAllAssets(ctx contractapi.
      TransactionContextInterface) ([]*Asset, error) {
// range query with empty string for startKey and
   endKey does an
// open-ended query of all assets in the chaincode
   namespace.
    resultsIterator, err := ctx.GetStub().
       GetStateByRange("", "")
    if err != nil {
      return nil, err
    }
    defer resultsIterator.Close()

    var assets []*Asset
    for resultsIterator.HasNext() {
      queryResponse, err := resultsIterator.Next()
      if err != nil {
        return nil, err
      }

      var asset Asset
      err = json.Unmarshal(queryResponse.Value, &asset
         )
      if err != nil {
        return nil, err
      }
      assets = append(assets, &asset)
    }

    return assets, nil
  }

  func main() {
    assetChaincode, err := contractapi.NewChaincode(&
       Record{})
    if err != nil {
      log.Panicf("Error creating record chaincode: %v"
         , err)
    }

    if err := assetChaincode.Start(); err != nil {
```

```
    log.Panicf("Error starting record chaincode: %v"
        , err)
  }
}
```