Plates2Block

A Hyperledger Fabric Implementation
for
Vehicle Plate Registry

March 11, 2020
Richard Qin, 101278511
Shermeen Kazi, 101270927
Estella Yeung, 100316780

Table of Contents

Group Project Background

Governments issue license plates to people and organizations to keep track of who owns a vehicle. Multiple parties need to access this information such as car dealers, insurance companies and the police. This requires a flawless and efficient design required for these parties to register, access, and update information. Currently, handling and registering license plate information means navigating paper and digital systems. This leads to duplication of data, increased risk of losing data or not having updated information.

For this project we would like to propose Plate2Block, a decentralized solution for license plate registry which would allow multiple organizations to accurately register and track vehicles.

Plates2Block would enable:

- Gathering vehicle information
- Gathering plate owner information
- Recording payment for license
- Recording assigned plate number
- Recording plate renewal data
- Query license plate to find owner and vehicle information

Plates2Block Business Model

This is a permissioned blockchain, so only registered users can access data. This would ensure that only authorized personnel are able to access Plate2Block. To prevent misuse and to generate income to support the system, there is a $10 charge for information access, $80 charge for new plate application, renewal or ownership change.

The following users can access the vehicle plate system:

- Owners of the vehicle plate can request for license registry to update, renew, change ownership, and access their own data with a password
- Car dealerships can apply for a license plate for a car buyer
- Government agencies can approve/decline requests to search, apply, update, renew, and ownership change
- Law enforcement can view
- Insurance agent can view license registry

Note:

Our user list has the government agency approving the owner's request to register, update, and change ownership. This means the chaincode should have the owner or other user of the system, such as Law Enforcement, creating a transaction to be approved by the government agency. If the transaction is

approved, the owner's vehicle plate can be registered, searched, updated, or ownership changed and the transaction for money is accepted. If the transaction is declined then payment is not accepted and no action can be taken.

However, the rest of our paper and our code will not reflect this, and will show the owners creating a transaction to register, update, renew, and change ownership without gaining agency approval.

The vehicle plate is an asset with the following attributes:

- Vehicle plate number: 12345678
- Vehicle VIN number: 1G3H5678901234567
- Vehicle owner name: Joe Blow
- Vehicle owner address: 2000-123 Alexander St., Any City, Any Country
- Insurance company name: Desjardin World Insurance Corp
- Insurance policy number: CB03062020
- Previous plate number: set to blank/space if new
- Previous owner name: set to blank/space if new
- Previous owner address: set to blank/space if new

Model Definition

The vehicle plate model definition:

"**Add new plate**": {

" Vehicle plate number": "integer",

"Vehicle VIN number": "string",

"Vehicle owner name": "string",

"Vehicle owner address": "string",

"Insurance company name": "string",

"Insurance policy number": "string"

" Previous plate number": "integer", (move existing record value to previous if ownership change, otherwise, set to blank)

"Previous owner name": "string", (see above)

"Previous owner address": "string"}

An instance of the **add new** vehicle plate:

{"12345678",
"1G3H5678901234567",
"Joe Blow",

“2000-123 Alexander St., Any City, Any Country”,
“Desjardin World Insurance Corp”,
“CB03062020”,
“ “,
“ “,
“ “}

On vehicle plate **search**, the user can submit one of the following search keys:

“**Search plate**”: {

“ Vehicle plate number”: “integer”,

OR

“Vehicle VIN number”: “string”}

An instance of **search** vehicle plate:

{“12345678”,
OR
“1G3H5678901234567”}

The vehicle plate **ownership update** definition. Update is generally on owner address or insurance information change:

“**Plate Ownership Update**”: {

“Vehicle plate number”: “integer”, (likely no change unless opt for a custom fancy plate number)

“Vehicle VIN number”: “string”, (should never change)

“Vehicle owner name”: “string”, (should not change in update)

“Vehicle owner address”: “string”, (can change in update)

“Insurance company name”: “string”, (can change in update)

“Insurance policy number”: “string” (can change in update)

“Previous plate number”: “integer”, (take the previous plate number if a new plate number is detected)

An instance of the vehicle ownership **update**:

{“88888888”,
“1G3H5678901234567”,

"Joe Blow",
"80 Ocean Blvd, Any City, Any Country", (can change)
"Best Insurance Corp", (can change in update)
"BB03112020", (can change in update)
"12345678"} (set to previous plate number if new plate number found)

The vehicle plate **ownership change** definition:

"**Plate Ownership Change**": {

"Vehicle plate number": "integer", (if plate number is different from data store, put the new number here and move what was stored into previous plate number)

"Vehicle VIN number": "string", (should never change)

"Vehicle owner name": "string", (the new owner)

"Vehicle owner address": "string", (the new owner)

"Insurance company name": "string",

"Insurance policy number": "string"

"Previous plate number": "integer", (on ownership change, new owner has the option of taking the existing vehicle plate number in the data store. If so, this field is set to blank in ownership change. Otherwise, move what is in the datastore previously into this slot)

"Previous owner name": "string",

"Previous owner address": "string"}

An instance of the vehicle ownership **change**:

{"87654321",
"1G3H5678901234567",
"Kitty Cat",
"80 Ocean Blvd, Any City, Any Country",
"Best Insurance Corp",
"BB03112020"
"12345678", (set to blank if new owner takes the previous owner plate number)
"Joe Blow",
"2000-123 Alexander St., Any City, Any Country"}

States in the Data Model

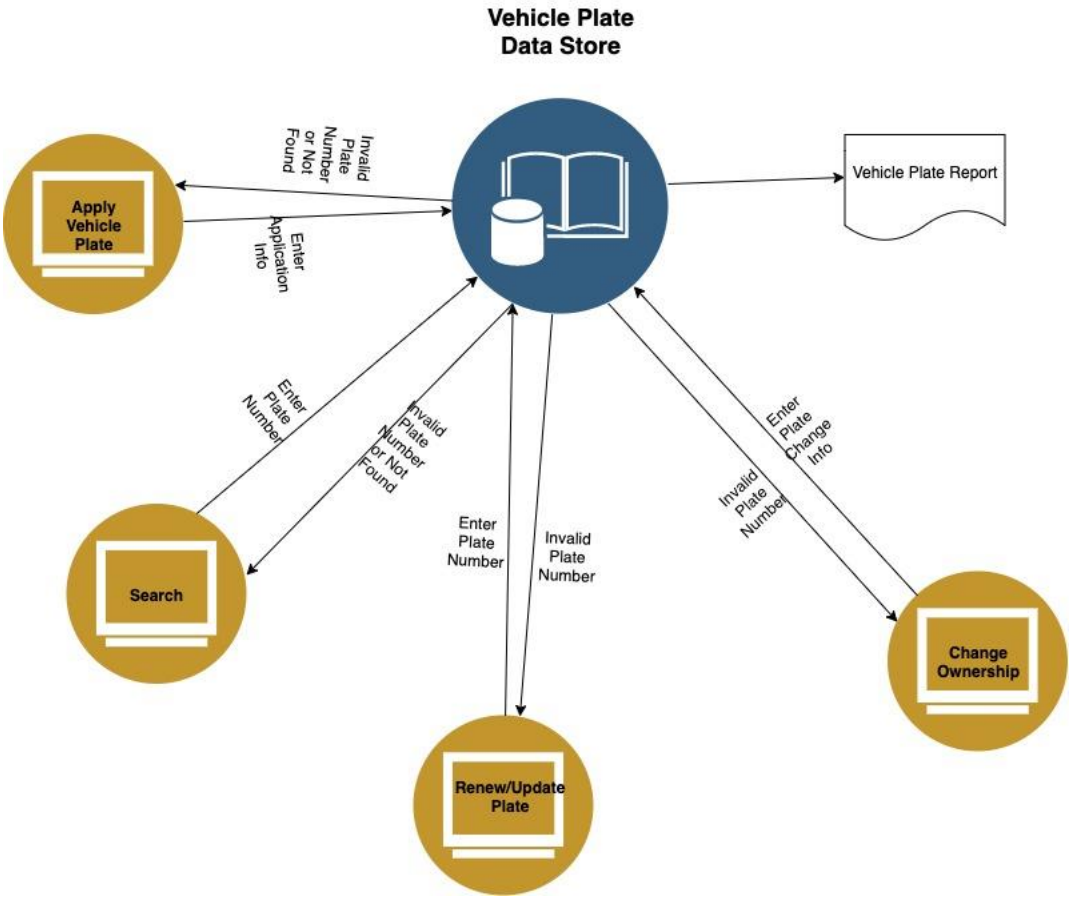
A state in data modelling represents a process. In the vehicle plate registry, there are the following states:

- Apply/Register/Add New vehicle plate

- Search vehicle plate
- Renew or update vehicle plate
- Transfer vehicle ownership
- Store vehicle plate data
- Produce vehicle plate report

Plates2Block State Diagram

Allow search plate
Gather vehicle plate owner info
Record assigned plate number
Record plate update, renewal, and ownership change data



From one state to another state, there are transitions. Here are the transitions identified for each of the state.

Transitions from Apply/Register/Add New Vehicle Plate

From Apply/Register/Add New Vehicle Plate to Store Vehicle Plate Data, the transitions are triggered by the following inputs:

- Plate number
  - generated, or a special owner defined unique number
  - Integer 8
- VIN number
  - a unique vehicle identifier from the manufacturer. In future, can validate this number against the car manufacturer's database to ensure accuracy
  - String
- Vehicle owner name
  - String
  - One line
- Vehicle owner address
  - String
  - One line. In future, can validate address against an address domain to ensure accuracy
- Insurance company name
  - String
  - One line; validate name
- Insurance policy number
  - String
  - One line. In future, validate policy number to ensure accuracy
- Clicking of a submit button
- System could return with the following error:
  - Plate number not unique, or
  - Insurance company name not found
  - Return to main selection screen
- Otherwise, Set process ID to **AddNew**

Transitions from Search Vehicle Plate

From Search Vehicle Plate to Store Vehicle Plate Data, the transitions are triggered by the following inputs:

- Plate number OR
  - Integer 8
  - Already exist in the system
- VIN number
  - String
  - Already exist in the system
- Clicking of a submit button
- System could return with the following error:
  - Plate number not found, or
  - VIN number not found

    - Return to main selection screen
- Otherwise, Set process ID to **Search**

Transitions from Renew or Update Vehicle Plate

In renew or update, users are not allowed to make name change. Record selection is on plate or VIN number. Owner may change the plate number to a user determined number. From Renew or Update Vehicle Plate to Store Vehicle Plate Data, the transitions are triggered by the following inputs:

- Plate number
    - Should be found in the system
    - Integer 8
- VIN number
    - Should be found in the system
    - String
- Vehicle owner name
    - Grey out; can't change; display from data store
    - One line
- Vehicle owner address: display from data store and allow user to change
    - String
    - One line
- Insurance company name: display from data store and allow user to change
    - String
    - One line
- Insurance policy number: display from data store and allow user to change
    - String
    - One line
- Previous Plate number
    - If a plate number is found, store the previous plate number here
    - Integer 8
- Clicking of a submit button
- System could return with the following error:
    - Plate number not found
    - Insurance company name not found
    - Return to main selection screen
- Otherwise, Set process ID to **RenewUpdate**

Transitions from Change Vehicle Ownership

From Change Vehicle Ownership to Store Vehicle Plate Data, the transitions are triggered by the following inputs:

- Vehicle plate number
  - Integer 8
  - Can be the number in data store or a new number from the new owner
- Vehicle VIN number
  - String
  - Should be in the system
  - If not found, print error message, stop processing
- Vehicle owner name: in ownership change, the new owner name is stored here
  - String
  - One line
- Vehicle owner address: in ownership change, the new owner address is stored here
  - String
  - One line
  - In future, address can be checked against a known address database to ensure accuracy
- Insurance company name: in ownership change, the new owner insurance info
  - String
  - One line
- Insurance policy number: in ownership change, the new owner insurance info
  - String
  - One line
- Previous plate number
  - Integer 8
  - Set to blank but if the new owner takes a new plate number, store the previous plate number here
- Previous owner name: in ownership change, the previous owner name is stored here
  - String
  - One line
- Previous owner address: in ownership change, the previous owner address is stored here
  - String
  - One line
- Clicking of a submit button
- System could return with the following error:
  - Plate number not unique, or
  - Insurance company name not found
  - Return to main selection screen
- Otherwise, Set process ID to **ChangeOwner**

From Store Vehicle Plate Data to Produce Vehicle Plate Report, the transitions are triggered by the following inputs:

- If process id = **AddNew**

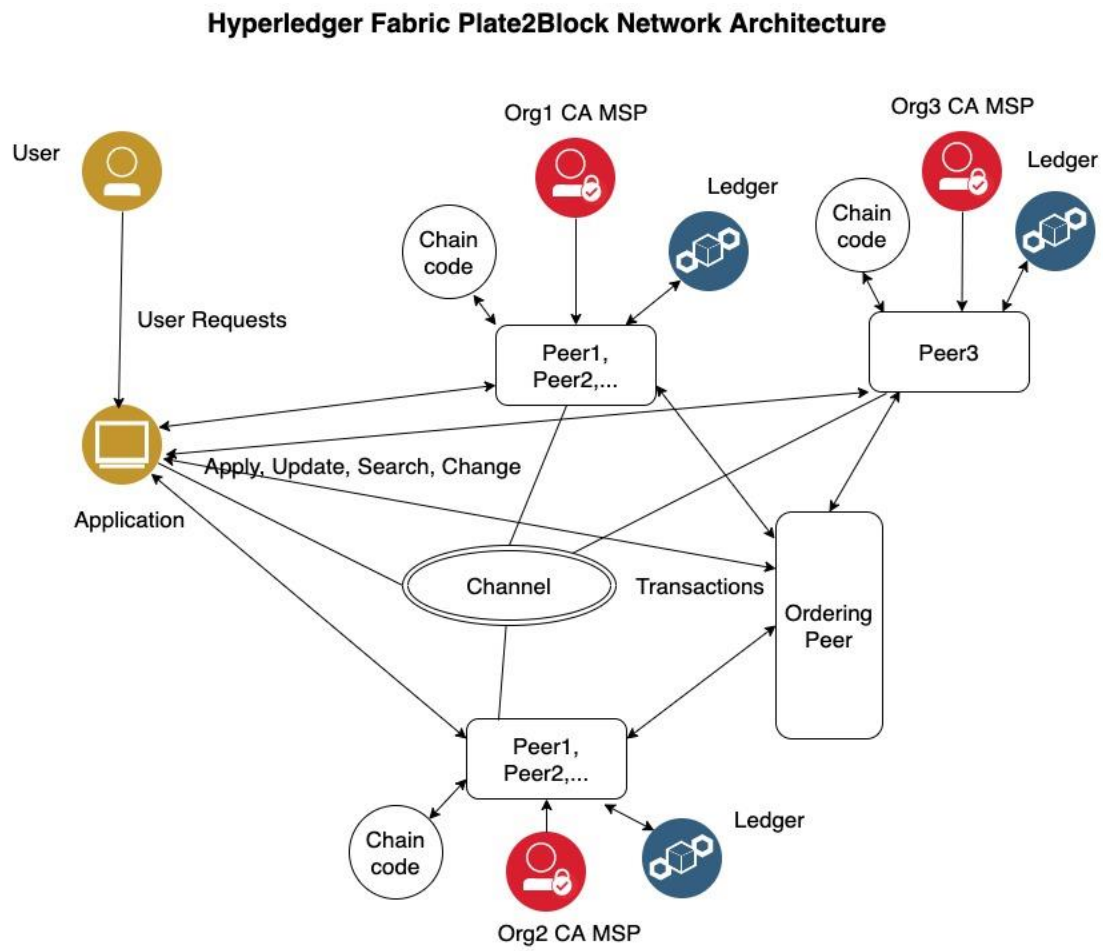
- Ensure plate number is unique and not used. If not, return error message ‘Plate number not unique’. Stop processing.
- Ensure insurance company exists. If not, return error message ‘Insurance company name not found’. Stop processing.
- Owner to confirm the screen display by pressing the submit button
- After user confirmation:
  - Print vehicle plate report
  - Display a message on screen ‘Vehicle Plate Application completed’
- Return to the main selection screen

- If process id = **Search**
  - Ensure plate number is found in the system. If not, return error message ‘Plate number not found’
  - Ensure the VIN number is found in the system. If not, return error message ‘VIN number not found’
  - User to confirm the screen display by pressing the submit button
  - After user confirmation:
    - Print vehicle plate report
    - Display a message on screen ‘Vehicle Plate search completed’
  - Return to the main selection screen

- If process id = **RenewUpdate**
  - Ensure plate number is found in the system. If not, return error message ‘Plate number not found’. Stop processing
  - Ensure insurance company exists. If not, return error message ‘Insurance company name not found’. Stop processing
  - Display the record to allow user update:
  - Plate number - owner has a chance to change the plate number. If not, grey out, can’t change
  - VIN number - grey out, can’t change
  - Vehicle owner name - grey out, can’t change
  - Vehicle owner address - can change
  - Insurance company name - as entered from the screen; ensure insurance company exists. If not, return error message ‘Insurance company name not found’. Stop processing
  - Insurance policy number - as entered from the screen
  - Owner to confirm the screen display by pressing the submit button
  - After user confirmation:
    - Print vehicle plate report
    - Display a message on screen ‘Renew/Update vehicle plate data completed’
  - Return to the main selection screen

- If process id = **ChangeOwner**
    - Ensure existing plate number is found in the system. If not, return error message ‘Existing plate number not found’. Stop processing
    - Ensure VIN number is found in the system. If not, return error message ‘VIN number not found’. Stop processing
    - Ensure new plate number is unique and not used. If not, return error message ‘Plate number not unique’. Stop processing
    - Ensure insurance company exists. If not, return error message ‘Insurance company name not found’
    - Owner to confirm the screen display of ownership change information by pressing the submit button
    - After user confirmation:
        - Print vehicle plate report
        - Display a message on screen ‘Vehicle Ownership Change completed’
    - Return to the main selection screen

Plates2Block Network Architecture



Peers

A blockchain network is comprised of peer nodes, each of which can hold copies of ledgers and copies of chaincodes (smart contracts). Because a peer is a host for ledgers and chaincodes, applications and administrators must interact with a peer if they want to access these resources. A peer is able to host more than one ledger.

- The Plates2Block network consists of peer1, peer2 and peer3, each of which maintains their own instances of the distributed ledger and instances of chaincodes. The peers use the same chaincode to access their copy of that distributed ledger
- Peers can be created, started, stopped, reconfigured, and even deleted. They expose a set of APIs that enable administrators and applications to interact with the services that they provide
- Chaincodes are used to query or update the peer's ledger instances
- Applications always connect to peers when they need to access ledgers and chaincodes

- The Software Development Kit (SDK) has APIs to enable applications connect to peers, invoke chaincodes to generate transactions, submit transactions to the network that will get ordered and committed to the distributed ledger, and receive events when this process is complete
- Through a peer connection, applications can execute chaincodes to query or update a ledger. The result of a ledger query transaction is returned immediately since all of the information required to satisfy the query is in the peer's local copy of the ledger. Peers never consult with other peers in order to respond to a query from an application

Certificate of Authority

The certificate of authority (CA) in hyperledger Fabric is an entity that issues digital certificates. CA provides identity registration function, certificate renewal and revocation function, and it is also a repository of public keys for verification. CA provides trusts to parties in the blockchain system.

In the Vehicle Plate Registry, each organization has peer1, peer2, and peer3. And there is a CA for each organization. The Orderers in the network has an ordererCA.

The role of the CA

- Every component has an identity issued by its organization's CA
- The component uses its identity to determine its organizational role
- This role determines the level of access it has to network resources such as read/write the ledger

Orderers

Orderers, aka, ordering nodes or ordering peers. They collect transactions from across the network into blocks and distribute the blocks to all peers. Orderers are involved in updates.

- Ledger updates
  - A more complex interaction between applications, peers and orderers
  - The application builds a transaction from all of the responses, and sends it to the ordering peer
  - The ordering peer collects transactions from across the network into blocks, and distributes these to all peers
  - The peer validates the transaction before applying to the ledger
  - Once the ledger is updated, the peer generates an event, received by the application, to signify completion
- Updates
  - Other peers must first consent to the change
  - Peers return to the application a proposed update — one that this peer would apply subject to other peers' prior agreement
  - The application sends the proposed updates to the entire network of peers as a transaction for commitment to their respective ledgers

- The orderers package transactions into blocks, and distribute them to the entire network of peers, where they can be verified before being applied to each peer's local copy of the ledger

Channels

Peers are connected using channels. Can set up a channel to allow a specific set of peers and applications to communicate with each other within the network. For example, an application can communicate directly with peer1 and peer2 using channel C - like a pathway for communications between particular applications and peers.

- In the vehicle plate registry current design, a channel is used to connect the peers. The existing business model does not require the setup of a specific pathway to communicate; nor does it require to setup a channel for private data

Membership Services Provider (MSP)

MSP is a component in the Hyperledger Fabric network that aims to offer an abstraction of a membership operation architecture (meaning it hides all but the relevant data about an object in order to reduce complexity and increase efficiency).

MSP is for internal system user roles, and access control definition. It is not for control of external users signing on the system through User Interface. MSP abstracts away all cryptographic mechanisms and protocols behind issuing and validating certificates, and user authentication.

An MSP may define their own notion of identity, and the rules by which those identities are governed (identity validation) and authenticated (signature generation and verification).

A Hyperledger Fabric blockchain network can be governed by one or more MSPs. This provides modularity of membership operations, and interoperability across different membership standards and architectures.

MSP configuration needs to be specified:

- Locally at each peer and orderer to enable peer, and orderer signing
- On the channels to enable peer, orderer, client identity validation, and respective signature verification (authentication) by and for all channel members.

Data Governance

Data governance is the overall management of confidentiality, availability, integrity, and security of data. Are there rules surrounding its use and protection? Data is the most important asset for a business and therefore, a need for a proactive approach.

The profiling of data is an important step toward data governance:

- Who owns it

- Where does it come from
- Where and how it is used
- Is there a description of the data
- Does it adhere to company policies and rules?
- What about government regulations on
  - Privacy
  - Quality assurance
  - The latest regulatory requirements

Then ensure the capability to find data:

- Where it is stored
- When it is archived
- What data protection is in place
- How data is cleansed

Governance At the System Level

- Organizations in a consortium agree to form a network using a set of policies
- A network's policies encode each organization's rights, like a constitution Examples:
  - Every organization has the same authority over the network
  - All organizations must agree any change to the network
  - A majority of organizations must agree any change to the network
  - A majority of organizations including particular organization must agree a change
- Transition between constitutions possible with agreement

Governance At the User Level

The approach to data governance in Plates2Block will be as follows:

- Provide documentation on data ownership, data description, where data comes from, and how it is used
- Input vehicle plate number is checked for uniqueness to ensure no number is assigned to more than one vehicle
- Input insurance company is checked for existence
- Input insurance policy number is required. And if the insurance companies join the network, that would allow verification of input VIN against the VIN in the policy to enhance accuracy (in future, if vehicle manufacturers put vehicle data on blockchain, it would allow owner entry of VIN against manufacturer's blockchain data)
- Owner birthdate is not required thus it eliminates the GDPR compliance requirement
- Blockchain data store is different from a centralized database because it is a decentralized data store with the same transactions recorded in each node across the network. Because of the consensus mechanism and record immutability, the blockchain data is much more trusted, almost impossible to have malicious attacks. Cyber security is ensured without the use of expensive security software and constant monitoring requirements

- There is a built-in Hyperledger Fabric network user access control with user enrollment and registration processes - a user must first be accepted into the system before it can access the system
- Data availability is 7/24 because there is no single point of failure. If one node is down, users can access data in another node of the network
- In the Vehicle Plate Registry design, we implemented a higher barrier of access by charging a per record search fee. This simple step greatly enhances the privacy of the registry data

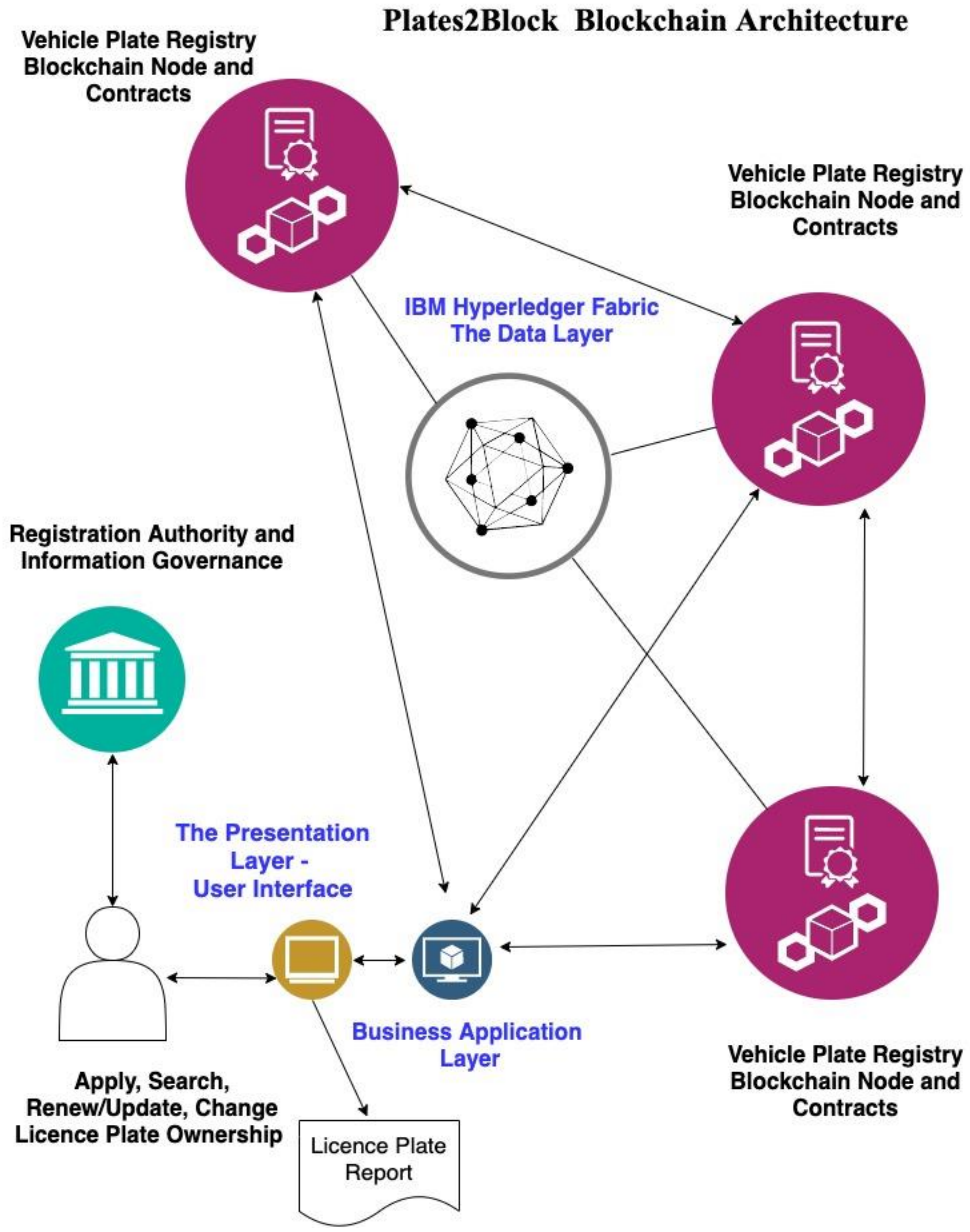
Solution Design

Plates2Block makes for a good blockchain use case as traditional license plate registry systems lack the coordination required to handle and access license plate information efficiently.

- Vehicle plate registration is an appropriate asset to be tracked using blockchain technology
- The participants are known because it is a permissioned blockchain
- There are rules around privacy and confidentiality because it is a private blockchain - the network can decide who joins, who can add/retrieve information; and there is a fee structure that can block potential misuse
- Transactions are endorsed through the specific endorsers in the network. After endorsement, the transactions are then validated by those on the network permissioned to receive them
- The network is governed by a consortium of members and policy makers to agree on a set of rules that include laws and regulations of the real-world system

The design for the registry system design is separated into 3 layers:

- The presentation layer
- The business application layer
- The data layer



The Presentation Layer

This layer acts as a go between the end user and the system - it functions as a shield to cover the complexity of the system from the user via a friendly user interface application. The user interface application takes the necessary inputs from the user, formats the input suitable for processing in the application layer. The presentation layer acts like a translator between the user and the system where data compression, decompression, encryption, decryption are completed in this layer.

User identification and registration takes place in this layer; this data is stored off-chain, not stored in the blockchain.

Language used to implement this layer can be node.js, React, JavaScript.

User profile of the Vehicle Plate Registration application:

- Car owners:
  - Enter the necessary information to apply for a car vehicle plate
  - Search for vehicle plate info
  - Update or renew existing vehicle plate info
  - Transfer or sale of the vehicle
  - In future, insurance policy is verified via QR code from policy owner's digital card
- Insurance Agency:
  - Search vehicle plate info when there is a car accident
  - In future, a digital insurance card can be issued to the car owner with a provable policy via QR code. This code can be checked into the blockchain
- Law Enforcement:
  - This registry is especially useful to Law Enforcement. This group of users will search the registry to find up-to-date information on car owners. They will pay to use the system and then charge back to car owners. Maybe in future it can be arranged with the registration authority that Law Enforcement to pay a yearly lump sum for unlimited search
  - Police can issue violations straight to drivers via license plate or VIN scan, even if not present
  - Authorities could also suspend driver's vehicle for failure to renew, or lapsed insurance in real-time
  - Drivers could be immediately notified via registration authority app synced to blockchain API
  - Payments for violations can be made to registration authority
- Any registered users:
  - Search vehicle plate info by paying a fee
- Registration Authority:
  - Internal registration and system portals are synced with blockchain to receive and update driver profiles with new ownership
  - Fees associated with registration can be paid digitally
  - License plates, tags and other physical paperwork is sent to address on file

Vehicle Plate Registration Workflow

- Owner prepays to the registration authority and obtains an access code to the vehicle plate registry system
- Vehicle owner activates the vehicle plate registration process

- The user interface system invokes business APIs, verifies insurance company name, policy number, and VIN (against an external VIN database. If not able to verify, generate an error message, return to the initial screen)
- On successful processing of the above step, the vehicle plate ownership can be registered to the car owner by the smart contract
- Print the vehicle plate report

Vehicle Plate Search Workflow

- User prepays to the registration authority and obtains an access code to the vehicle plate registry system
- User enters the vehicle plate, or VIN number
- The user interface system invokes business APIs, checks if either one of the number is found in the system
- If plate number not found, returns with 'Plate number not found'. Stop processing
- If VIN number not found, returns with 'VIN number not found'. Stop processing
- Print the vehicle plate report

Vehicle Plate Renewal/Update Workflow

- Owner prepays to the registration authority and obtains an access code to the vehicle plate registry system
- Vehicle owner activates the renewal or update vehicle plate process
- The user interface system invokes business APIs, verifies insurance company name, policy number, and VIN. If not able to verify, generate an error message, return to the initial screen
- On successful processing of the above step, the vehicle plate is renewed or updated by the smart contract
- Print the vehicle plate report

Vehicle Ownership Change Workflow

- Owner prepays to the registration authority and obtains an access code to the vehicle plate registry system
- Vehicle owner activates the change ownership process that includes both existing and new owner
- New owner can retain existing vehicle plate number or generate a new number
- New owner enters owner registration information
- The user interface system invokes business APIs, verifies insurance company name, policy number, and VIN. If not able to verify, generate an error message, stop processing

- On successful processing of the above step, the vehicle plate ownership can be transferred to the new owner by the smart contract
- Print the vehicle plate report

The Business Application Layer

This is the App Server area with programs written in JavaScript containing integration logics to call chaincodes (smart contracts) and to relate back and forth with the user interface applications.

The Data Layer

This is the Hyperledger Fabric network layer with defined structures to manage the blockchain. Functions of this layer are already established and defined. The business application layer interacts with the data layer via chaincodes. The following is a highlight of functions of this layer.

- Peers
  - A Hyperledger Fabric network comprises a set of nodes
  - The most common type of node is a peer
  - Peers are owned by different organizations
  - Organizations can own any number of nodes (2 or 3 is common)
  - They maintain ledger and state
  - Commit transactions
  - May hold smart contract (chaincode)
- Endorsing Peers to provide endorsement
  - A client sends transactions to the peers specified by the endorsement policy
  - Each transaction is then executed by specific peers and its output is recorded (this is the endorsement step)
  - After execution, transactions enter the ordering phase
- Ordering Peers to provide ordering service
  - The ordering service manages multiple channels
  - On every channel, it provides the following services:
    - Atomic broadcast for establishing order on transactions
    - Implementing the broadcast and deliver calls
    - Reconfiguration of a channel
  - Ordering peers batch transactions received from the atomic broadcast and form blocks
  - A block is cut as soon as one of three conditions is met:
    - The block contains the specified maximal number of transactions
    - The block has reached a maximal size (in bytes)
    - An amount of time has elapsed since the first transaction of a new block was received
- Certificate of Authority
  - Registration of identities
  - Certificate renewal and revocation
  - A repository of public keys for verification
- Member Services Provider

    - Maintains the identities of all nodes in the system (clients, peers, and OSNs)
    - Uniquely identifies the identities and roles for a channel
    - Is responsible for issuing node credentials that are used for authentication and authorization
    - In permissioned blockchain, all interactions among nodes occur through messages that are authenticated, typically with digital signatures
    - The membership service comprises a component at each node, where it may authenticate transactions, verify the integrity of transactions, sign and validate endorsements, and authenticate other blockchain operations
    - Tools for key management and registration of nodes are also part of the MSP
- Ledgers
    - All peers maintain the blockchain ledger
    - An append-only data structure recording all transactions in the form of a hash chain
- World state data store
    - The peer transaction manager (PTM) maintains the latest state in a versioned key-value store
    - The PTM uses a local key-value store with implementations using LevelDB and Apache CouchDB
- Chaincodes
    - Program code that implements the application logic and runs during the execution phase
    - It is the central part of a distributed application in Fabric
    - Special chaincodes exist for managing the blockchain system and maintaining parameters, collectively called system chaincodes
- Wallets
    - Wallets store the identities available to a component
    - Applications can use multiple wallets & identities
    - Allows applications connect to gateway with identity
- Channels
    - Channels connect peers
    - The channel defines the scope of the network
    - Each channel has a single ledger that is replicated to every peer that is a member of it
    - Can have a private channel if necessary