CSE446: Blockchain & Cryptocurrencies

Lecture – 19: HF - 2

Inspiring Excellence

Agenda

- Hyperledger Fabric

HF: Modularity

- A pluggable *ordering service* establishes allows different types of consensus algorithms to be integrated with HF
- A pluggable *membership service provider* is responsible for associating entities in the network with cryptographic identities
- Smart contracts ("chaincode") run within a container environment (e.g. Docker) for isolation
  - They can be written in standard programming languages
- The ledger can be configured to support a variety of DBMSs
- A pluggable endorsement and validation policy enforcement that can be independently configured per application

HF: privacy & confidentiality

- Hyperledger Fabric, being a permissioned platform, enables confidentiality through its channel architecture
- Basically, participants on a Fabric network can establish a “channel” between the subset of participants that could grant visibility to a particular set of transactions
  - Think of this as a sub-network
- Thus, only those nodes that participate in a channel have access to the smart contract (chaincode) and data transacted, preserving the privacy and confidentiality of both

HF: performance and scalability

- Fabric is quite scalable because of its permissioned model of achieving consensus
- Performance of Fabric is reported to be quite satisfactory
  - Around 3500 tx/s (https://arxiv.org/pdf/1801.10228v1.pdf)
  - Compare this with only 3-5 tx/s for bitcoin and around 10 tx/s for Ethereum with PoW

HF: identity

- Fabric relies on a concrete identity management architecture
- HF provides a membership identity service that manages user IDs and authenticates all participants on the network
- Access control lists can be used to provide additional layers of permission through authorisation of specific network operations
  - For example, a specific user ID could be permitted to invoke a chaincode application, but be blocked from deploying new chaincode

Fabric membership services

- Hyperledger Fabric underpins a network where all participants have known identities
- Public Key Infrastructure (PKI) is used to generate cryptographic certificates which are tied to organisations, network components, and end users or client applications
- As a result, data access control can be manipulated and governed on the broader network and on channel levels
- This “permissioned” notion of Hyperledger Fabric, coupled with the existence and capabilities of channels, helps address scenarios where privacy and confidentiality are of paramount concerns

Fabric CA and PKI

- Remember: users on a blockchain network are represented using public keys
- For any private blockchain, we need **verifiable** identities
- It means, there must be a way to guarantee that a certain public key represents an authorised entity in a network
- Based on this assurance we can create rules which will dictate who can participate in the blockchain network

https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd

Fabric CA and PKI

- But, how can we ensure this association?
- We use a Trusted Third Party (**TTP**) which vouches that a certain public key belongs to a certain entity
- In security, such TTPs are known as Certificate Authorities (**CAs**)

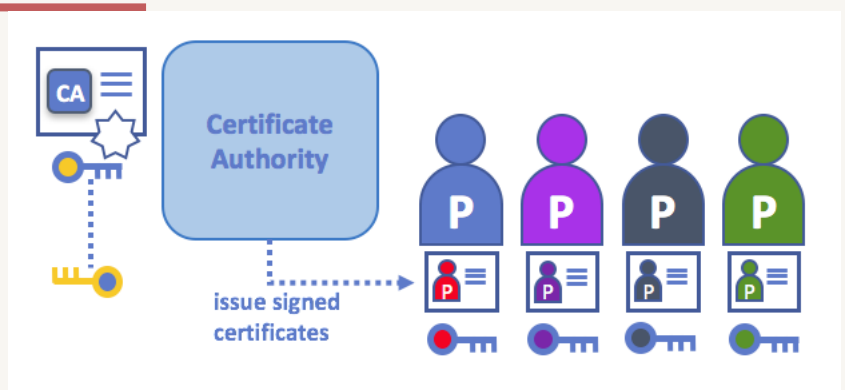
https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd

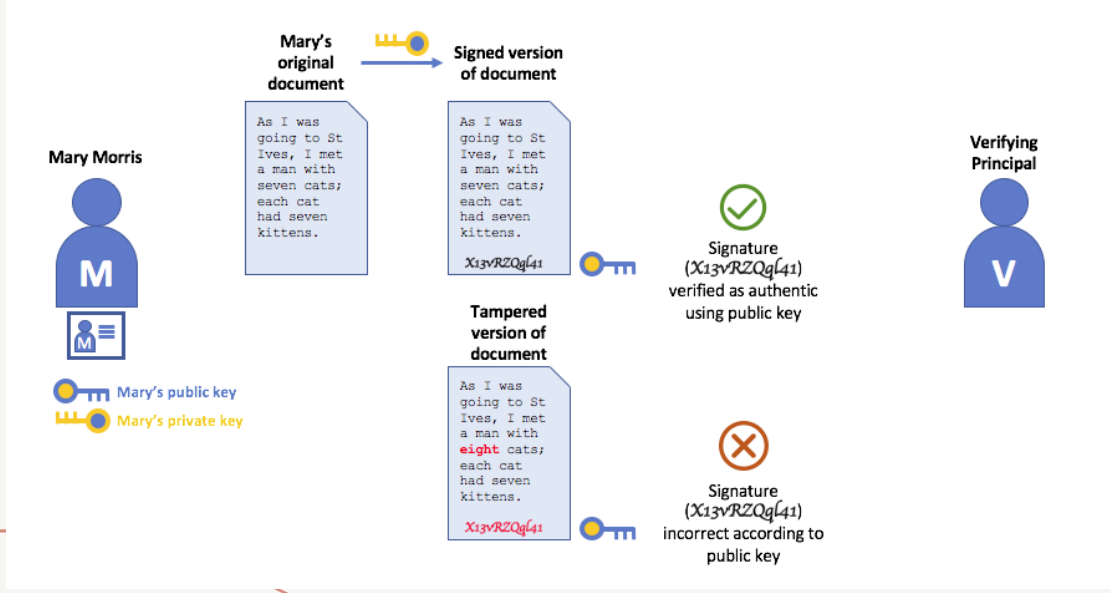
Fabric CA and PKI

- A CA issues a digital certificate which is used to bind a public key with an entity
- This association looks something like this:
  - "I, Mango CA, certify that Mr. Rahim belongs to Organisation O4 has this public key P_k and this is valid till 19 December, 2023"

Fabric CA and PKI

- Every digital certificate is then signed by the private key of the CA
- Anybody can utilise the public key of the CA to verify the signature in the certificate
- Once verified, everybody trusts the association between the public key and the entity
- If the certificate is tampered, the signature verification will fail





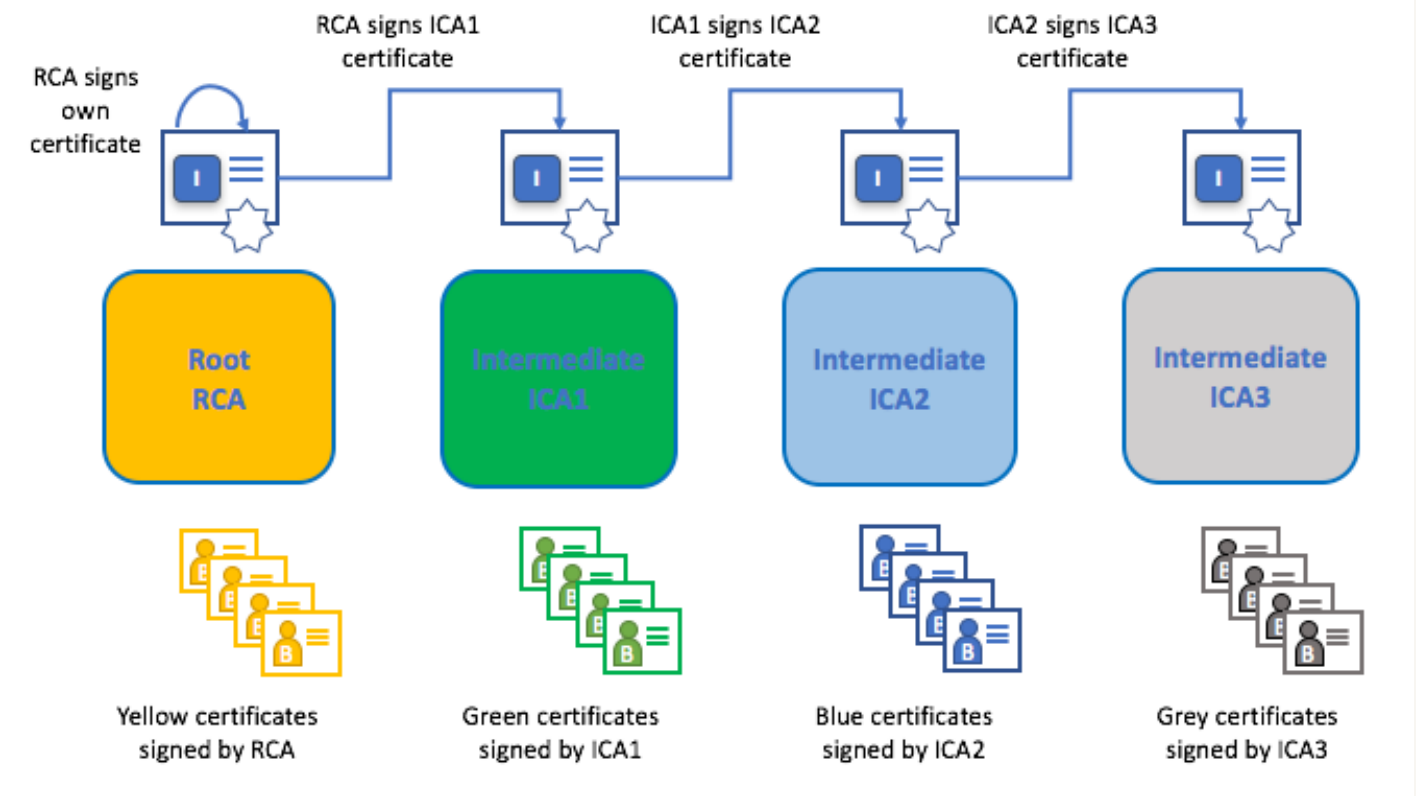
Fabric CA and PKI

- But, the problem is how do we get the public key of the CA to verify the signature in the certificate?
  - We used the signed certificate to get the public key of an entity
  - Does it mean we will need another certificate to get a CA's public key?
  - Who will sign such certificates? How will the signature verification work?
  - It is a circular problem!

Fabric CA and PKI

- Solution:
  - Create a hierarchy: Root CAs and Intermediate CAs
- Intermediate CAs distribute certificates to millions of users
- Root CAs distribute certificates for the intermediate CAs
- Embed the public keys of Root CAs to the OS and devices
  - In order to verify the root CA certificates
- Such a hierarchy is known as **PKI** (Public Key Infrastructure)

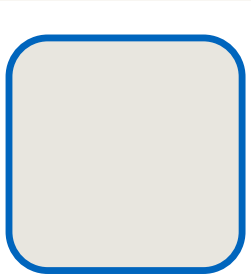
Fabric network: CA and PKI

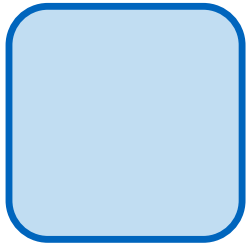


Fabric CA is a private root CA provider capable of managing digital identities of Fabric participants that have the form of **X.509** certificates for the Fabric network

Nodes and roles

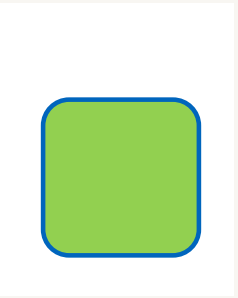
- Committing Peer
  - Maintains the state of the Blockchain and commits transactions
  - It verifies endorsements (rules and policies) and validate the results of a transaction
- Endorsing Peer
  - An endorsing peer is always also a committing peer
  - The difference is that an endorsing peer additionally takes transaction proposals (explained later) and executes them to create endorsements (explained later)



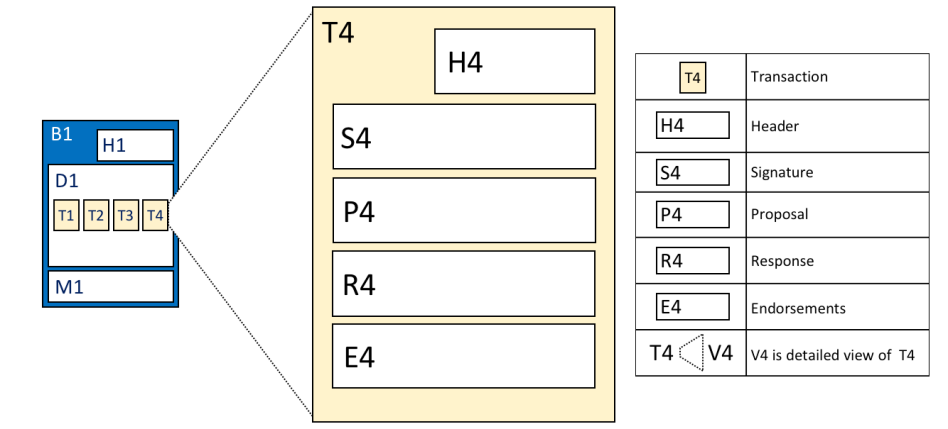


Nodes and roles

- Ordering service node
  - Applications must submit transactions to an ordering service node (Orderer)
  - The node then collects transactions and orders them sequentially
  - The transactions are then put into a new block and delivered to all peers of a specific channel
  - Does neither hold ledger nor chaincode

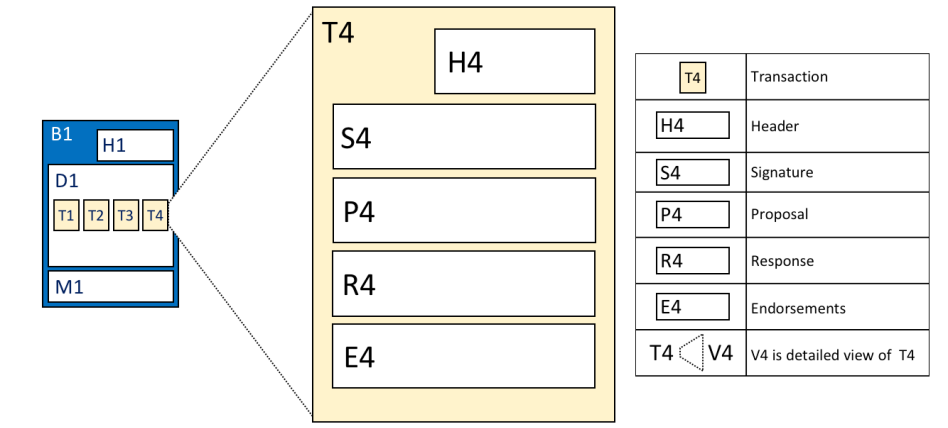


Fabric network: transactions



- Header: This section, illustrated by H4, captures some essential metadata about the transaction
  - for example, the name of the relevant chaincode, and its version
- Signature: This section, illustrated by S4, contains a cryptographic signature, created by the client application (user)
  - This field is used to check that the transaction details have not been tampered with, as it requires the application's private key to generate it
- Proposal: This field, illustrated by P4, encodes the input parameters supplied by an application to the smart contract which creates the proposed ledger update

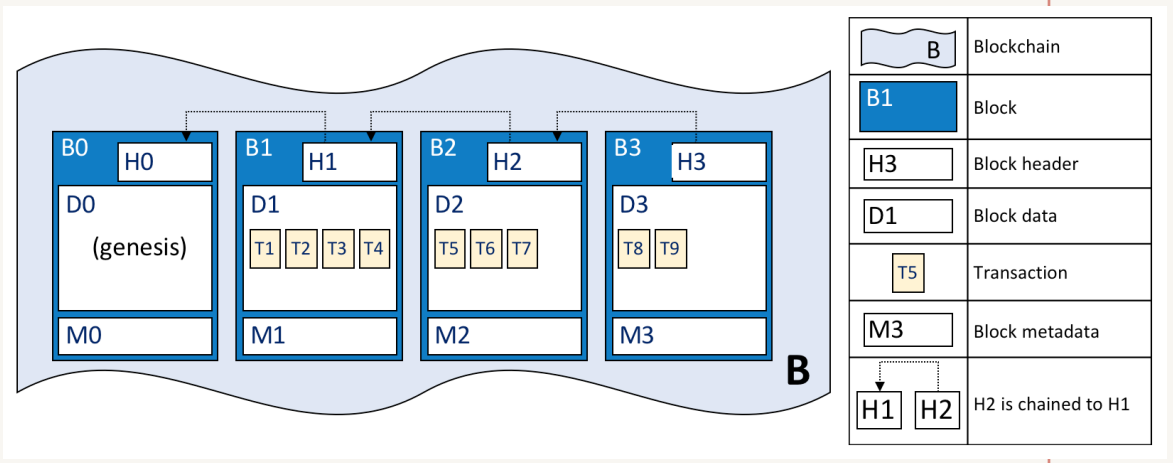
Fabric network: transactions



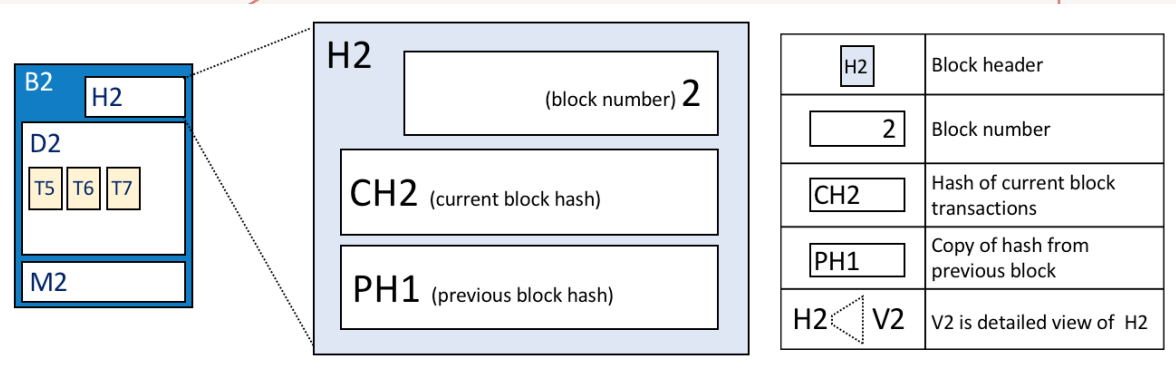
- Response: This section, illustrated by R4, captures the before and after values of the world state, as a Read Write set (RW-set)
  - It is the output of a smart contract
  - If the transaction is successfully validated, it will be applied to the ledger to update the world state
- Endorsements: As shown in E4, this is a list of signed transaction responses from each required organisation sufficient to satisfy the endorsement policy

Fabric network: blocks

- A blockchain B containing blocks B0 – B3
- B0 is the first block in the blockchain, the genesis block
- We can see that block B2 has a block data D2 which contains all its transactions: T5, T6, T7
- B2 has a block header H2
  - which contains a cryptographic hash of all the transactions in D2 as well as with the equivalent hash from the previous block B1
- This creates the chain among the blocks

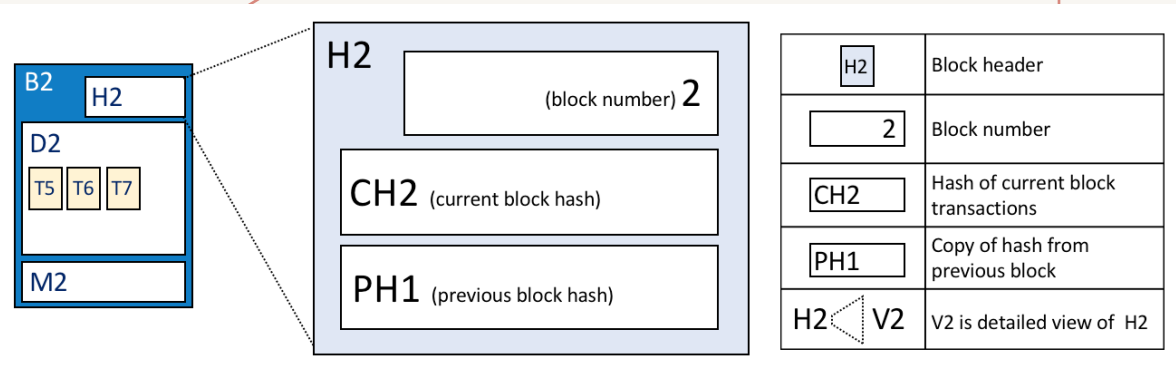


Fabric network: blocks



- Block Header: This section comprises three fields, written when a block is created
  - Block number: An integer starting at 0 (the genesis block), and increased by 1 for every new block appended to the blockchain
  - Current Block Hash: The hash of all the transactions contained in the current block
  - Previous Block Hash: A copy of the hash from the previous block in the blockchain

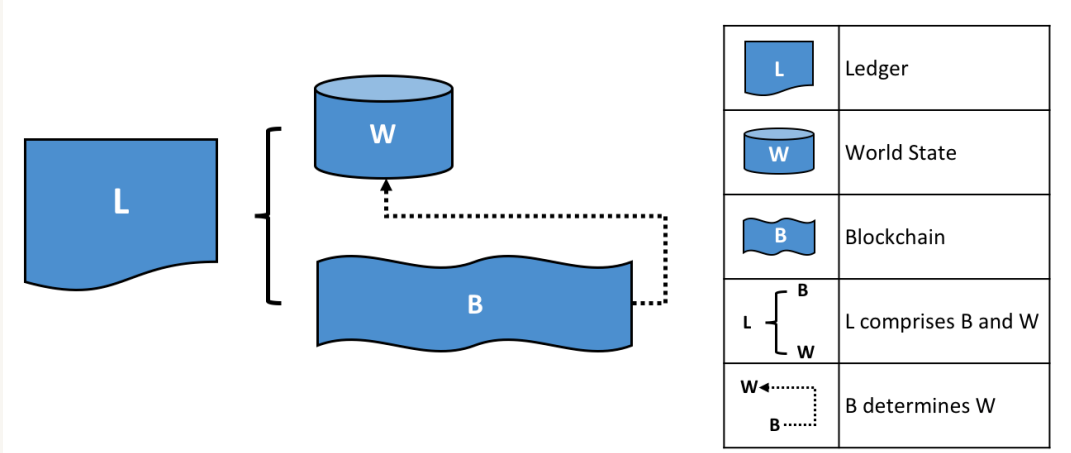
Fabric network: blocks



- Block Data
  - This section contains a list of transactions arranged in order
  - It is written when the block is created by the ordering service (block creation service)
- Block Metadata
  - This section contains the time when the block was written, as well as the certificate, public key and signature of the block writer
- Such metadata also contains a valid/invalid indicator for every transaction

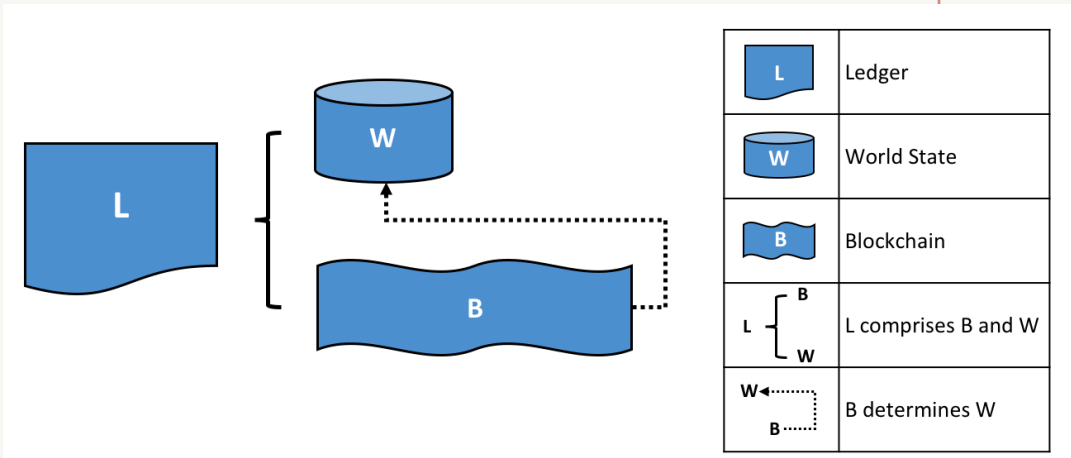
Fabric network: ledger

- In Hyperledger Fabric, a ledger consists of two distinct, though related, parts
  - a world state and a blockchain
- Each of these represents a set of facts about a set of business objects
- A **world state** is a database that holds a cache of the **current values** of a set of ledger states



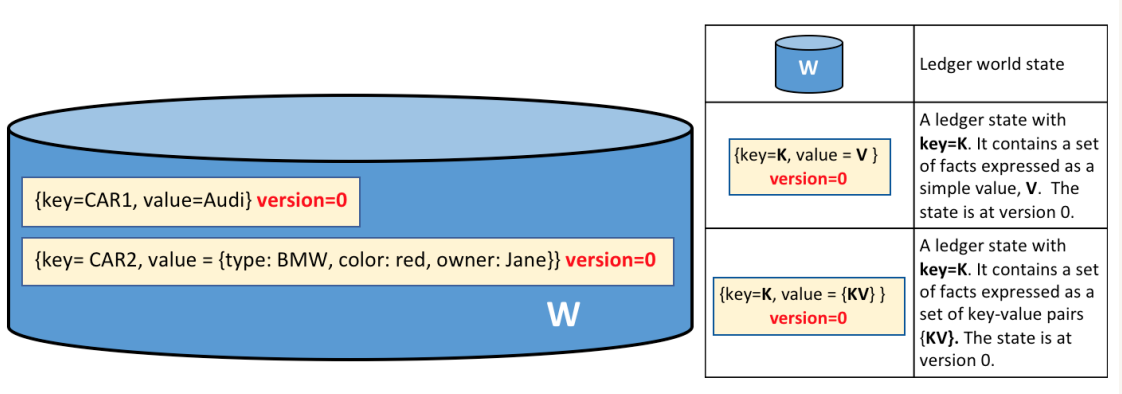
Fabric network: ledger

- The world state makes it easy for a program to directly access the current value of a state
- That is, you do not need to retrieve/calculate the state by traversing the entire transaction log as in Bitcoin
- Ledger states are, by default, expressed as **key-value** pairs



Fabric network: ledger

- A ledger world state might contain two types of states
- The first state is: key=CAR1 and value=Audi
- Here, the value is simple
- The second state has a more complex value:
- key=CAR2 and value={model:BMW, color=red, owner=Jane}
- Both states are at version 0
- The version number is incremented as the states are updated

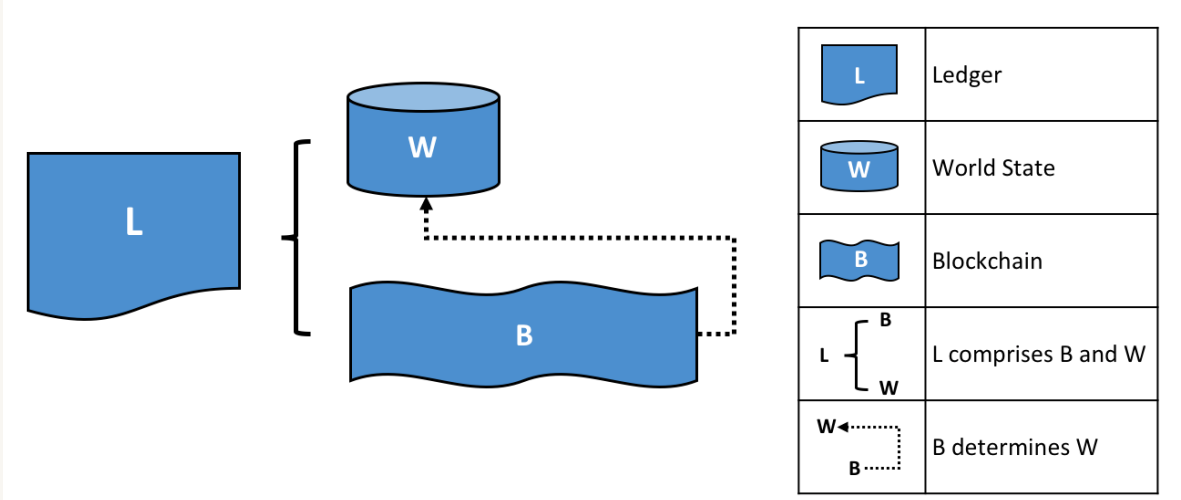


Fabric network: ledger

- At the initial stage, when a ledger is first created, the world state is empty
- When a transaction is created and it is then recorded in the blockchain
  - The world state is updated and recorded in the database

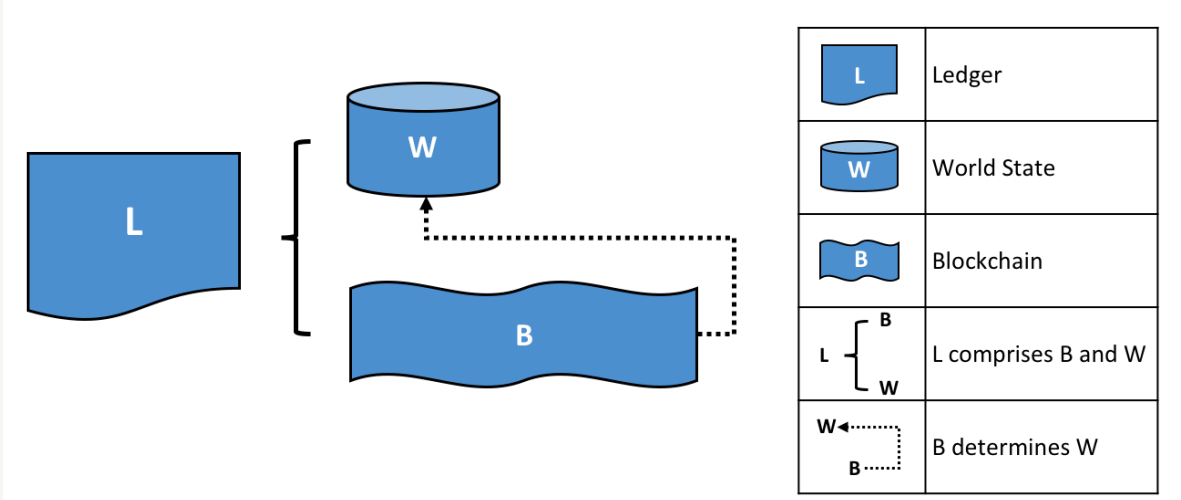
Fabric network: ledger

- The second part is the **blockchain**
- A blockchain is a transaction log that records all the changes that have resulted in the current the world state
- Transactions are collected inside blocks that are appended to the blockchain
- This enables you to understand the history of changes that have resulted in the current world state



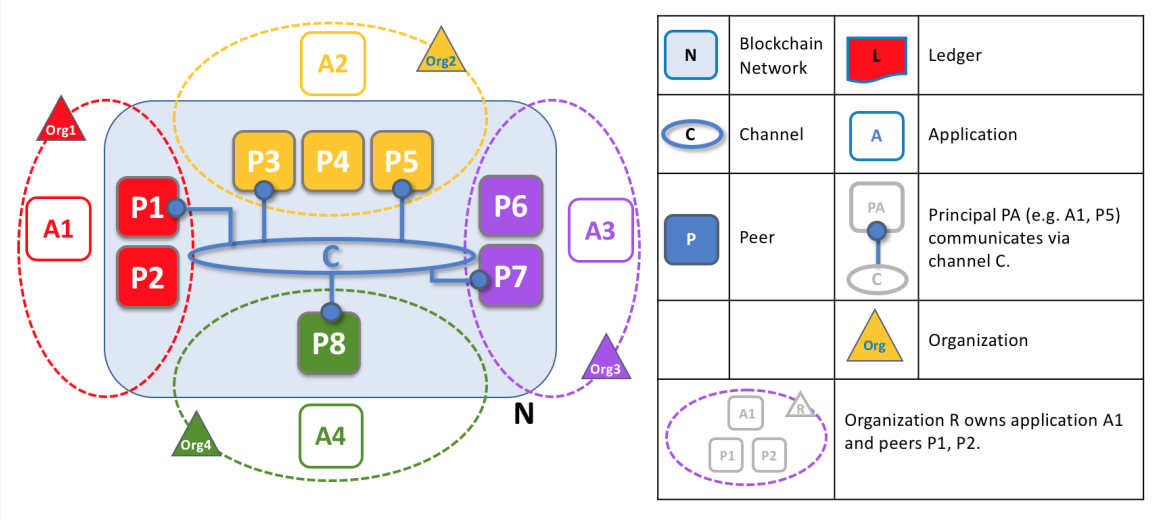
Fabric network: ledger

- The blockchain data structure is very different to the world state because once written, it cannot be modified
  - it is **immutable**
- We can also say that world state W is derived from blockchain B



Fabric network

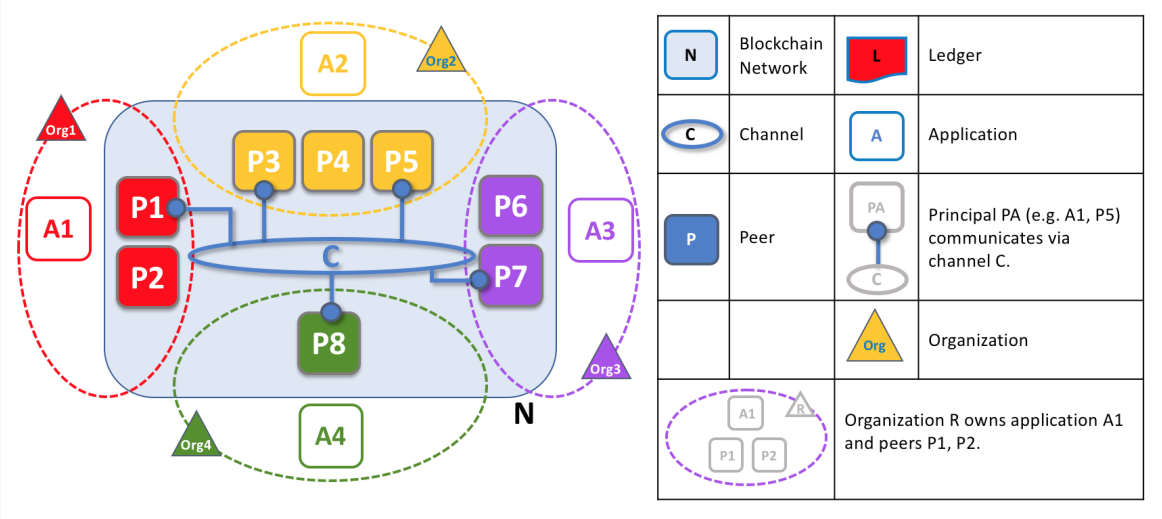
- In this example, we see four organisations contributing eight peers to form a network
- The channel C connects five of these peers in the network N – P1, P3, P5, P7 and P8
- The other peers owned by these organisations have not been joined to this channel, but are typically joined to at least one other channel



Peers in a blockchain network with multiple organisations

Fabric network

- Applications that have been developed by a particular organisation will connect to their own organisation's peers as well as those of different organisations
- For simplicity, an orderer node is not shown in this diagram



Peers in a blockchain network with multiple organisations

Question?