Hyperledger Fabric Key Concepts

최광훈 박사 / 아주대학교

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

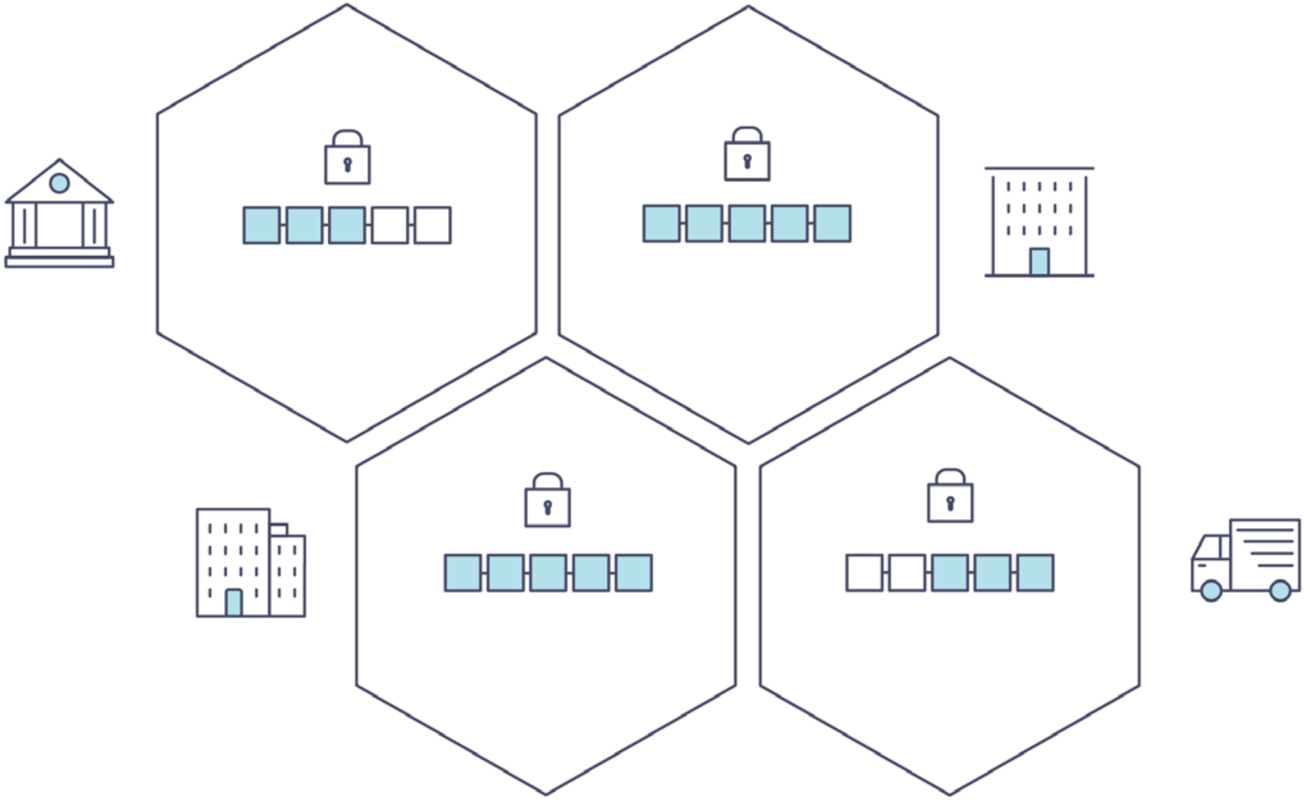
Ledger

The Ordering Service

Private data

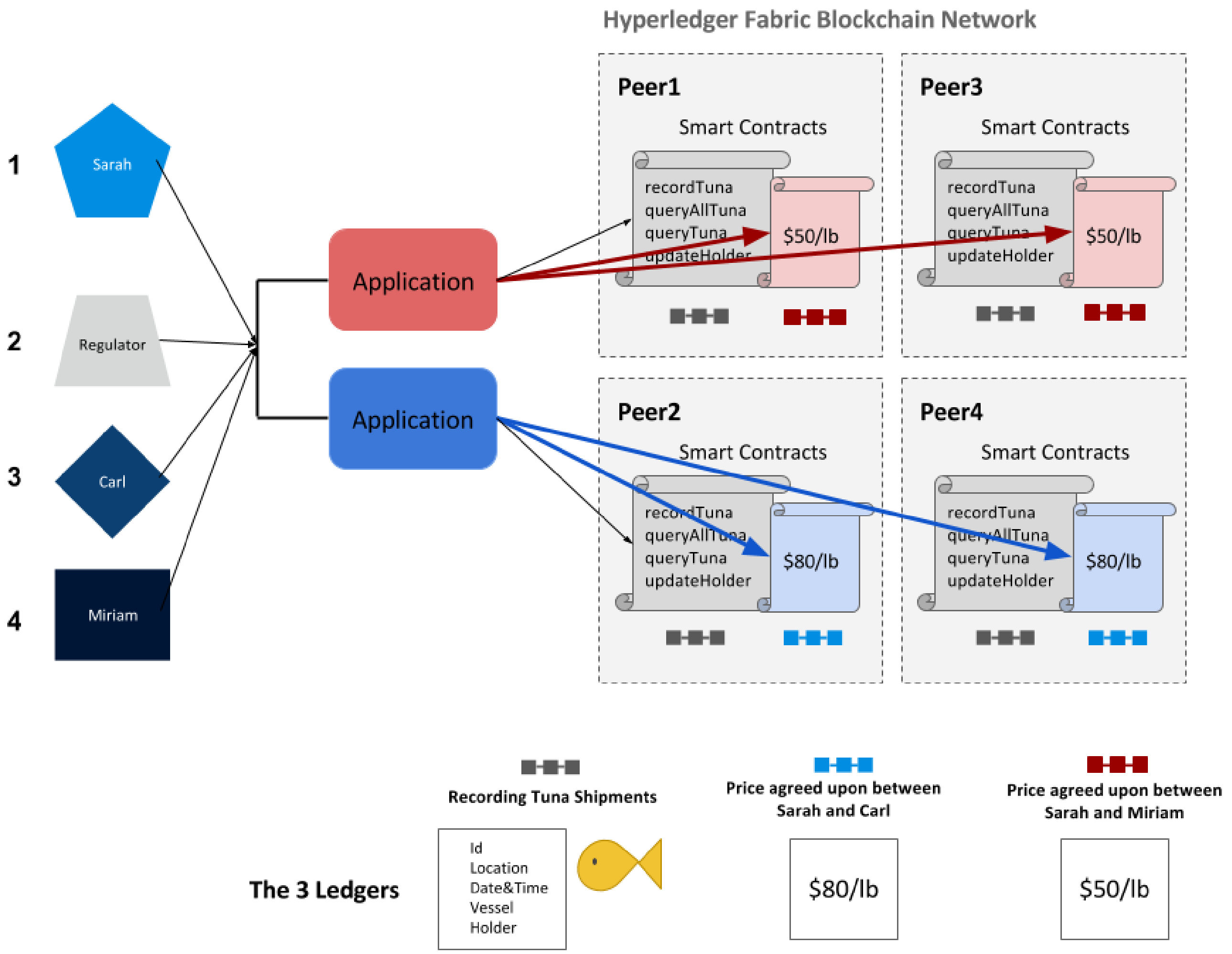
개요 - A Distributed Ledger

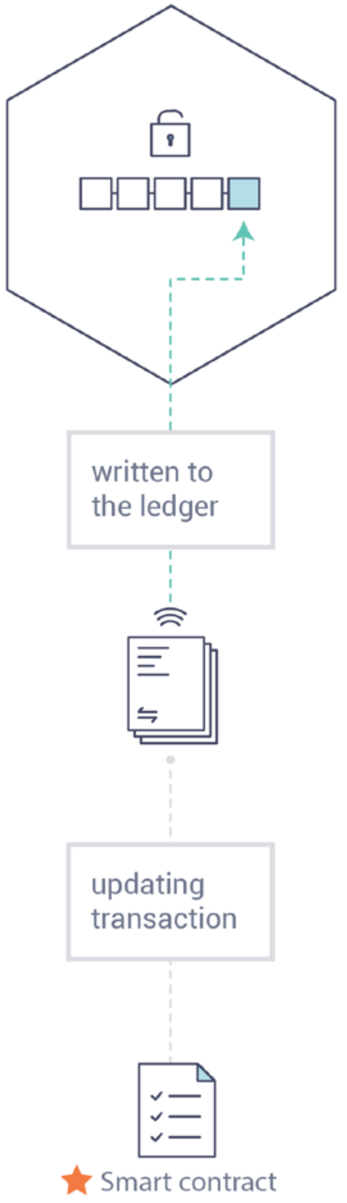
- blockchain network is a distributed ledger



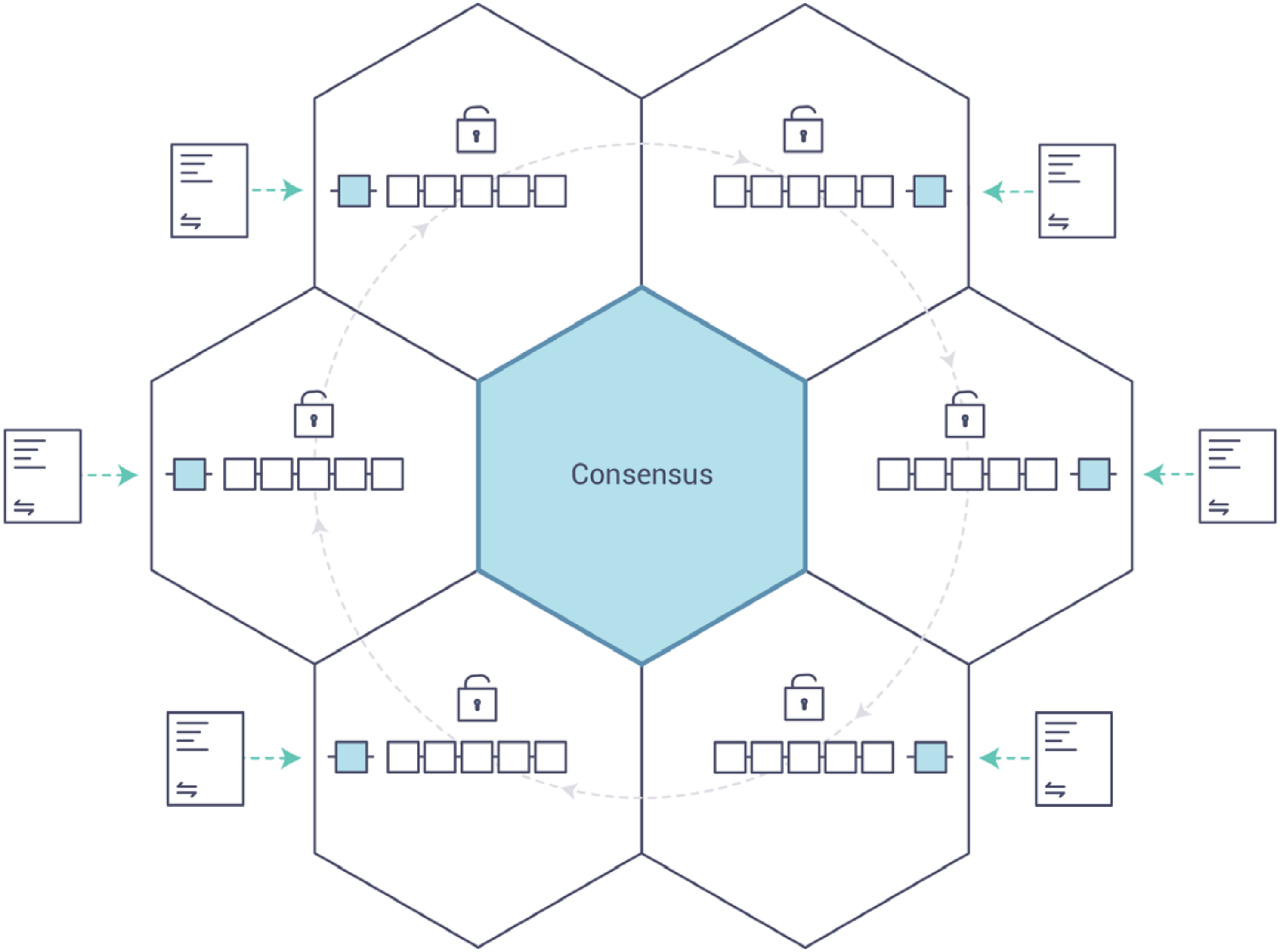
개요 - Smart Contracts

- provide controlled access to the ledger



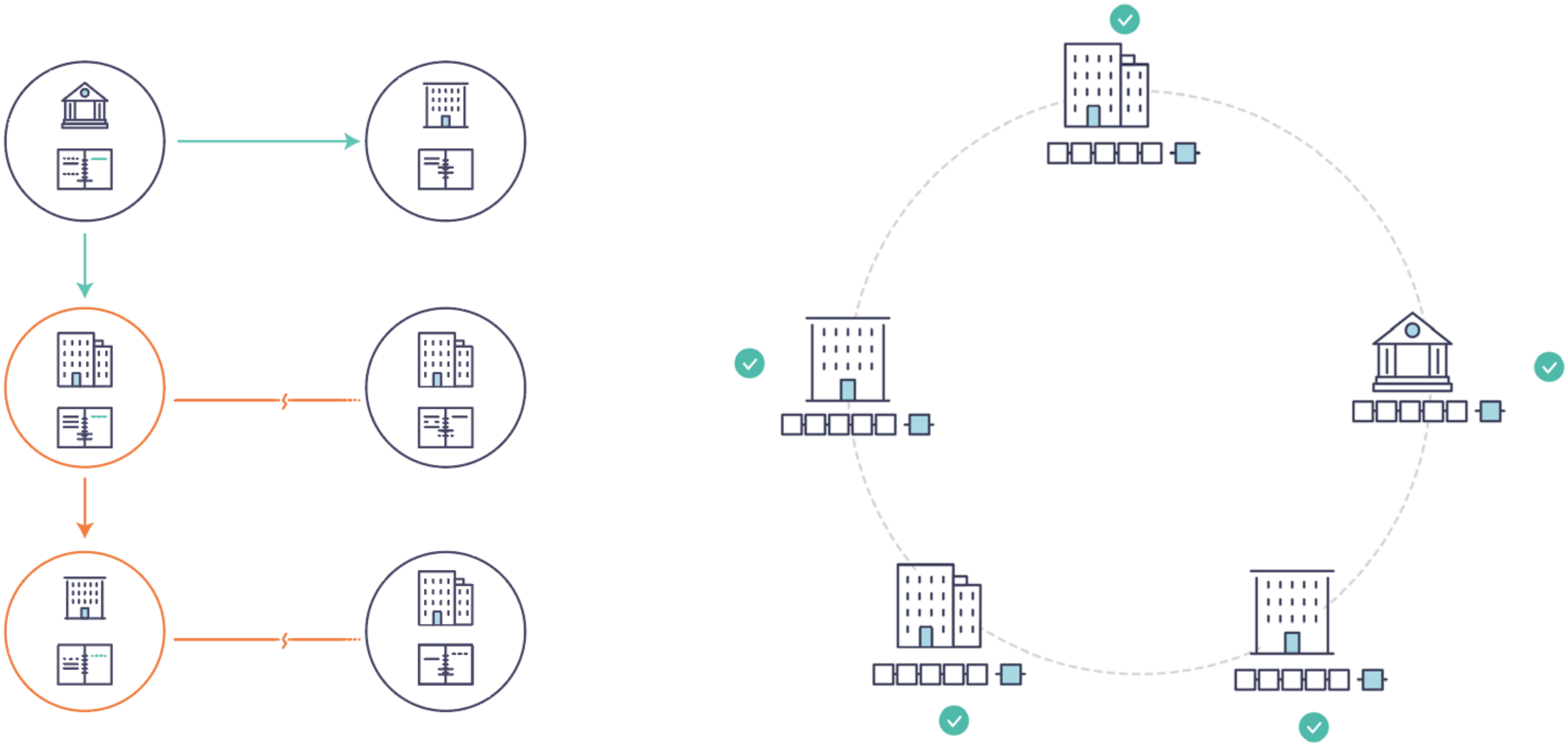


개요 - Consensus



WHY?

- 비지니스 네트워크 current vs blockchain



Hyperledger Fabric

- The Linux Foundation founded the Hyperledger project in 2015
- collaborative approach
- Hyperledger Fabric is one of the blockchain projects within Hyperledger
- **private** and **permissioned**
- the members of a Hyperledger Fabric network enroll through a trusted **Membership Service Provider (MSP)**

Shared Ledger

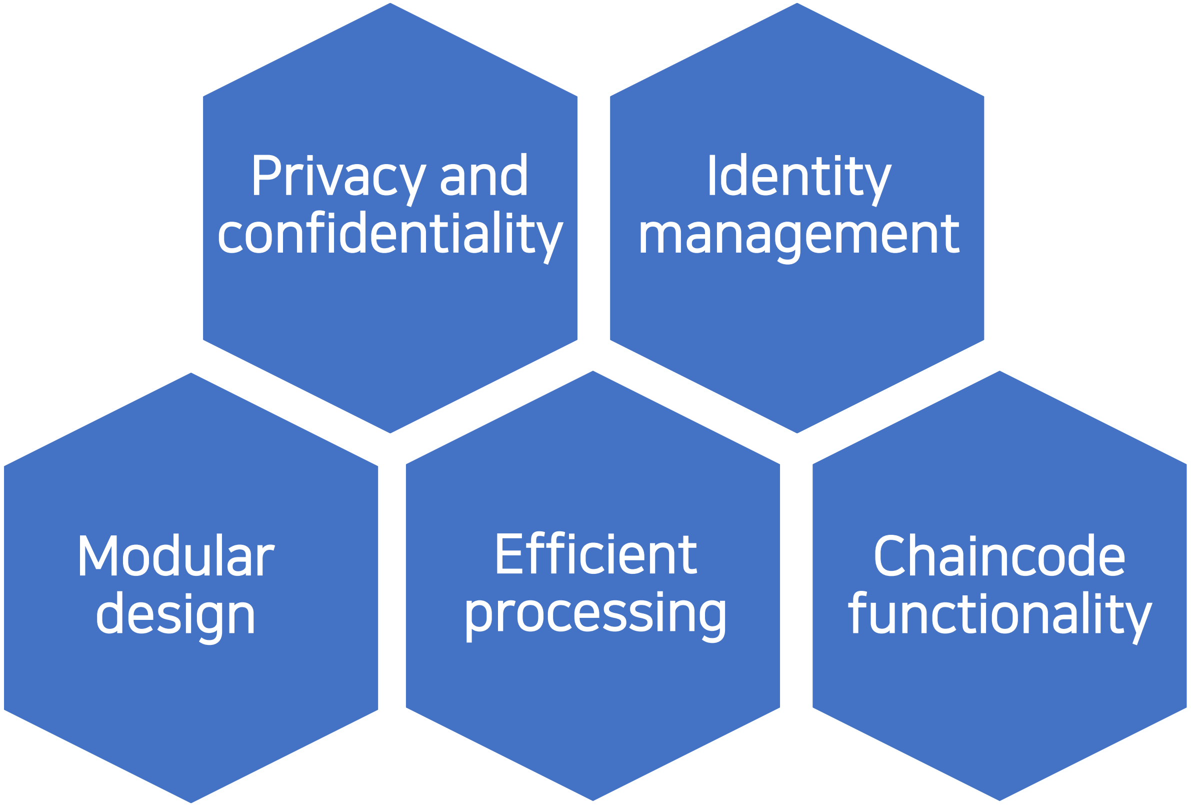
Smart Contracts

Privacy

Consensus

Fabric 기능

- Hyperledger Fabric is an implementation of distributed ledger technology (DLT)



Privacy and confidentiality

Identity management

Modular design

Efficient processing

Chaincode functionality

Fabric 모델

Assets

Chaincode

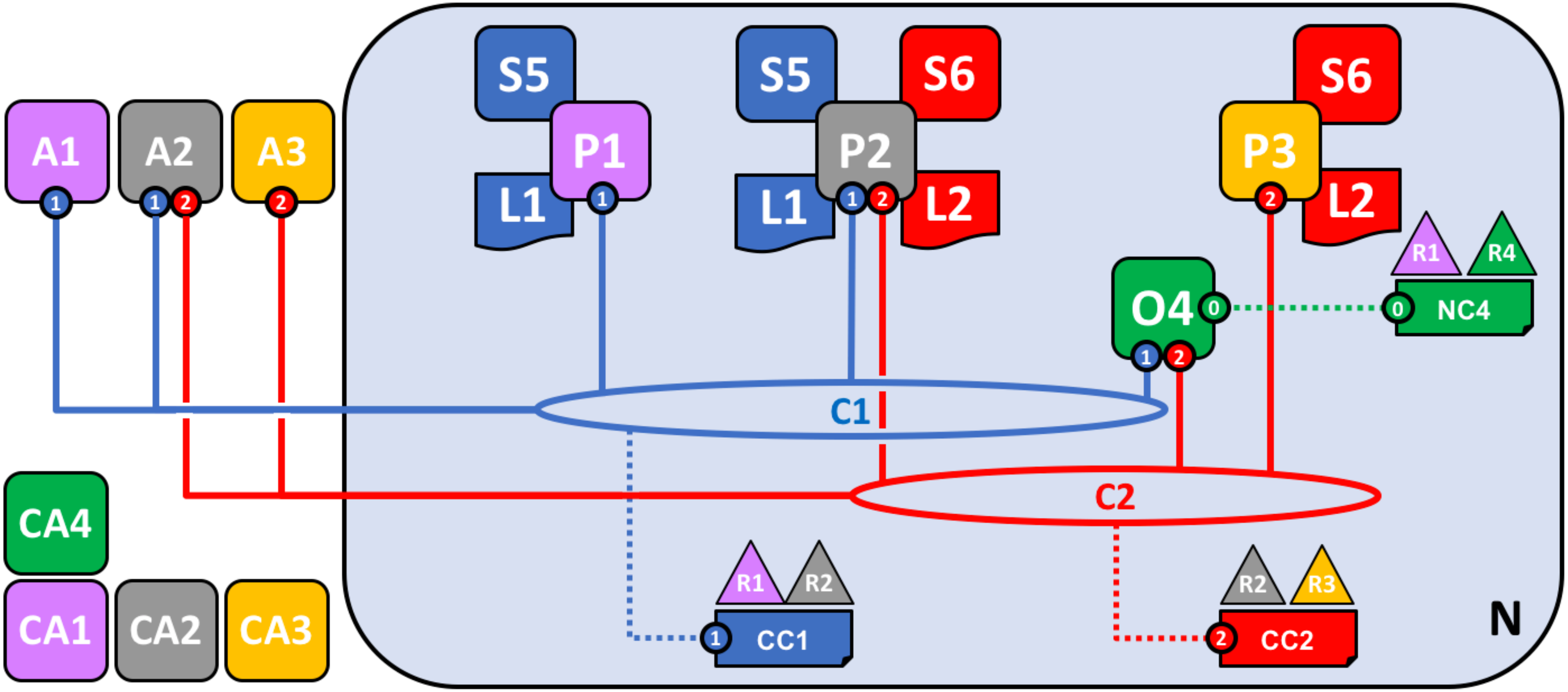
Ledger Features

Privacy

Security & MSP

Consensus

블록체인 네트워크


S5
S5
S6
S6
A1
A2
A3
P1
P2
P3
L1
L1
L2
L2
1
1
2
2
1
1
2
2
R1
R4
O4
0
0
NC4
1
2
C1
C2
CA4
CA1
CA2
CA3
R1
R2
R2
R3
1
CC1
2
CC2
N

Identity

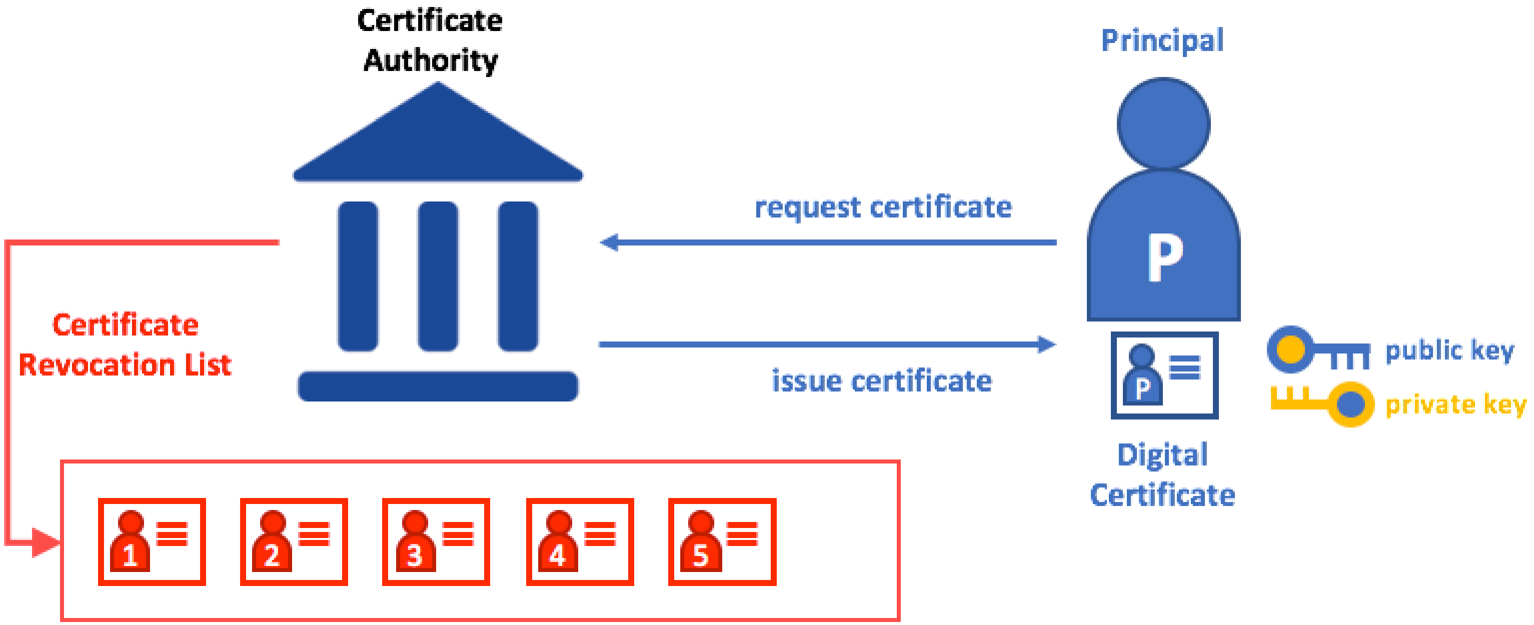
- *valid credit card is not enough*
- *a PKI provides a list of identities*

Accepted
Here

What are PKIs?

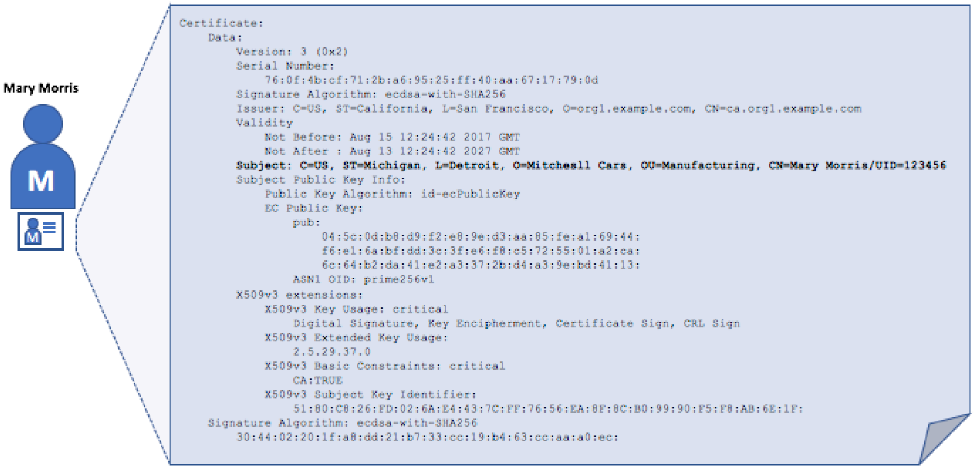
- A **public key infrastructure (PKI)** is a collection of internet technologies that provides secure communications in a network
- HTTPS

- Digital Certificates
- Public and Private Keys
- Certificate Authorities
- Certificate Revocation Lists



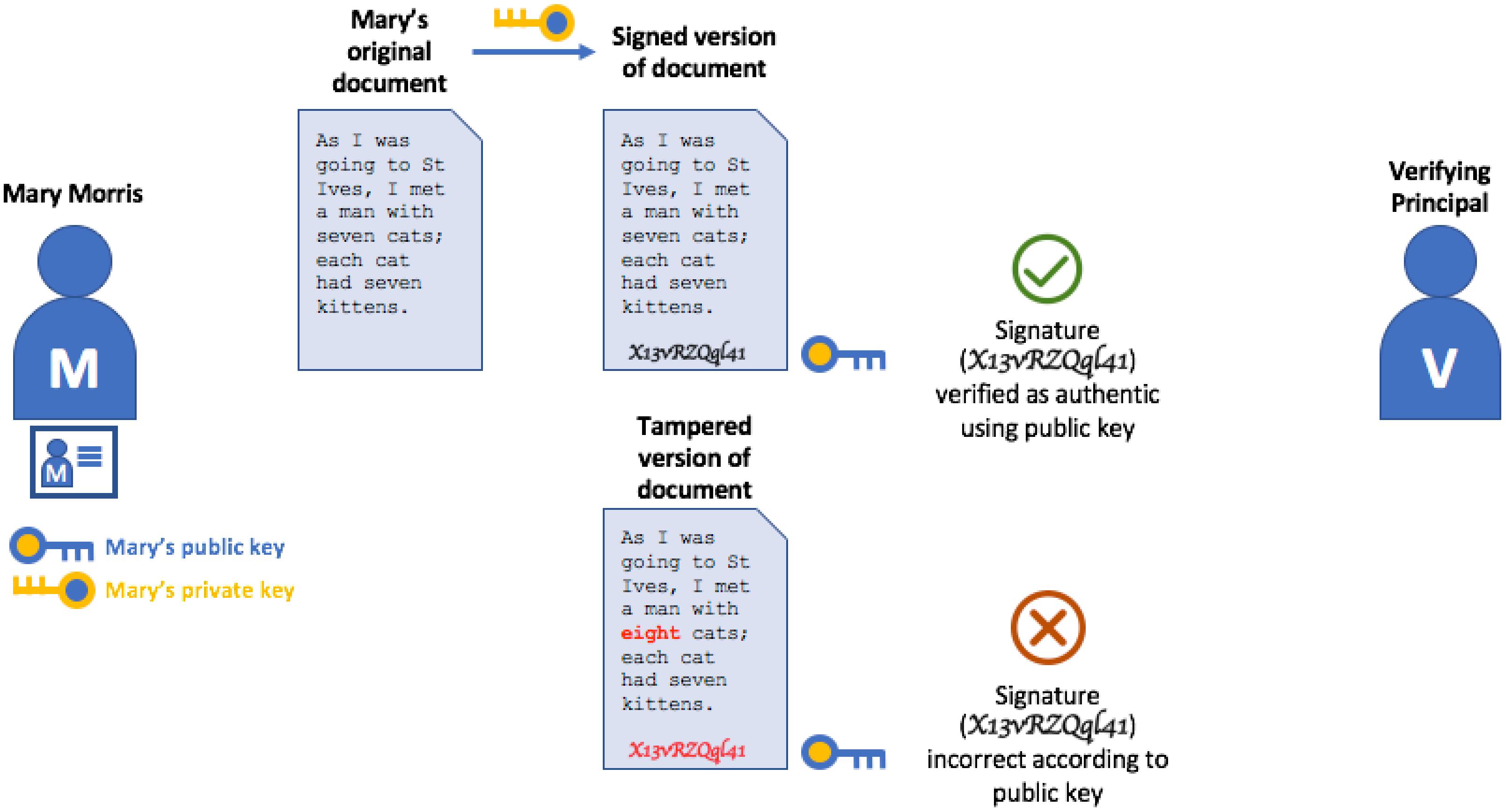
Digital Certificates

- A digital certificate is a document which holds a set of attributes relating to the holder of the certificate
- X.509 standard
- Mary Morris in the Manufacturing Division of Mitchell Cars in Detroit, Michigan might have a digital certificate
- SUBJECT attribute of C=US, ST=Michigan, L=Detroit, O=Mitchell Cars, OU=Manufacturing, CN=Mary Morris /UID=123456.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            76:0f:4b:cf:71:2b:a6:95:25:ff:40:aa:67:17:79:0d
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=US, ST=California, L=San Francisco, O=org1.example.com, CN=ca.org1.example.com
        Validity
            Not Before: Aug 15 12:24:42 2017 GMT
            Not After : Aug 13 12:24:42 2027 GMT
        Subject: C=US, ST=Michigan, L=Detroit, O=Mitcheall Cars, OU=Manufacturing, CN=Mary Morris/UID=123456
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
            EC Public Key:
                pub:
                    04:5c:0d:b8:d9:f2:e8:9e:d3:aa:85:fe:a1:69:44:
                    f6:e1:6a:bf:dd:3c:3f:e6:f8:c5:72:55:01:a2:ca:
                    6c:64:b2:da:41:e2:a3:37:2b:d4:a3:9e:bd:41:13:
                ASN1 OID: prime256v1
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Extended Key Usage:
                2.5.29.37.0
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                51:80:C8:26:FD:02:6A:E4:43:7C:FF:76:56:EA:8F:8C:B0:99:90:F5:F8:AB:6E:1F:
    Signature Algorithm: ecdsa-with-SHA256
        30:44:02:20:1f:a8:dd:21:b7:33:cc:19:b4:63:cc:aa:a0:ec:
```

공개키 개인키 인증



As I was going to St Ives, I met a man with seven cats; each cat had seven kittens.

As I was going to St Ives, I met a man with seven cats; each cat had seven kittens.

X13vRZQql41

Signature (X13vRZQql41) verified as authentic using public key

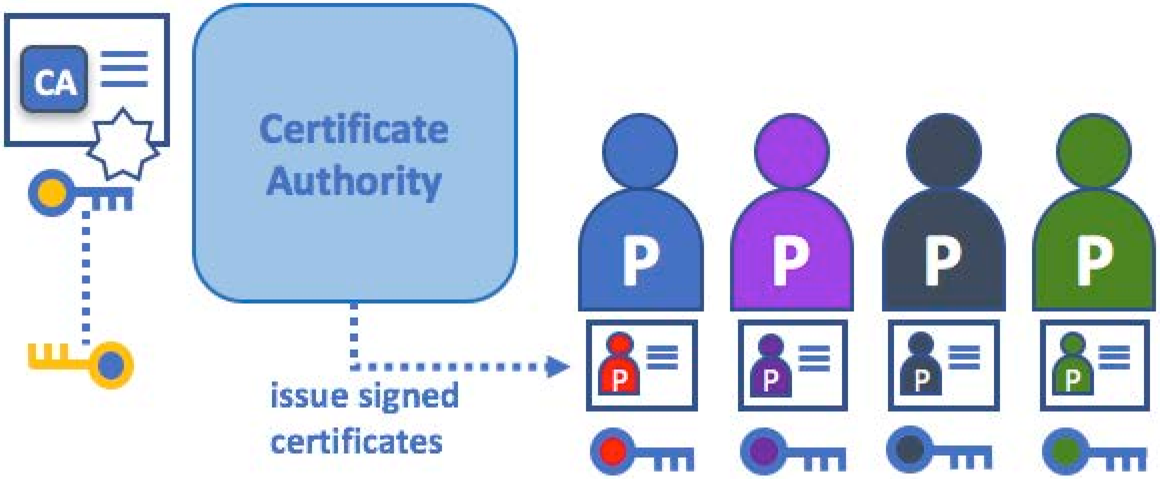
As I was going to St Ives, I met a man with eight cats; each cat had seven kittens.

X13vRZQql41

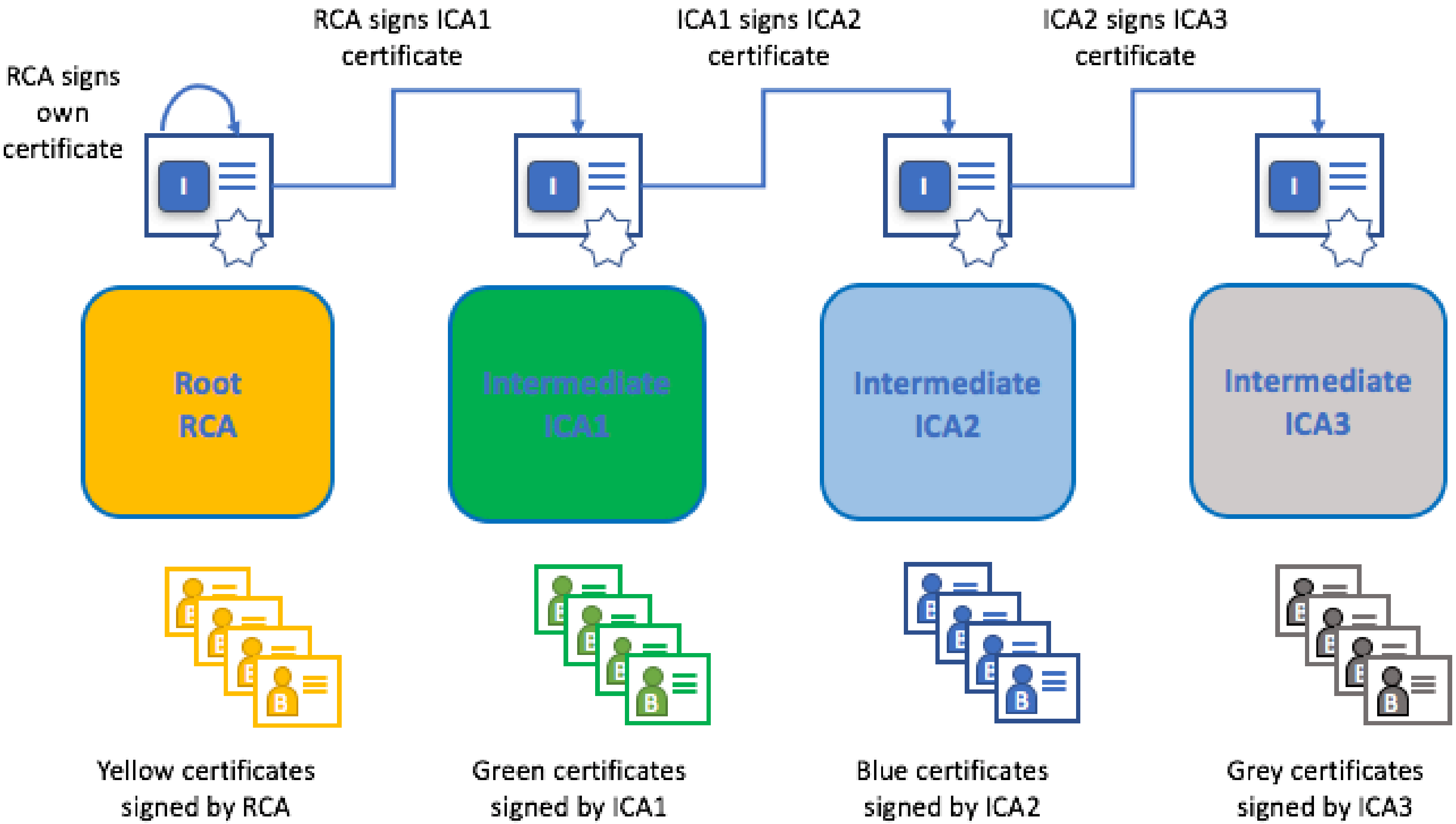
Signature (X13vRZQql41) incorrect according to public key

인증기관

- CAs are a common part of internet security protocols
- Symantec (originally Verisign), GeoTrust, DigiCert, GoDaddy, and Comodo, among others.



Root CAs, Intermediate CAs



RCA signs own certificate

RCA signs ICA1 certificate

ICA1 signs ICA2 certificate

ICA2 signs ICA3 certificate

Root RCA

Intermediate ICA1

Intermediate ICA2

Intermediate ICA3

Yellow certificates signed by RCA

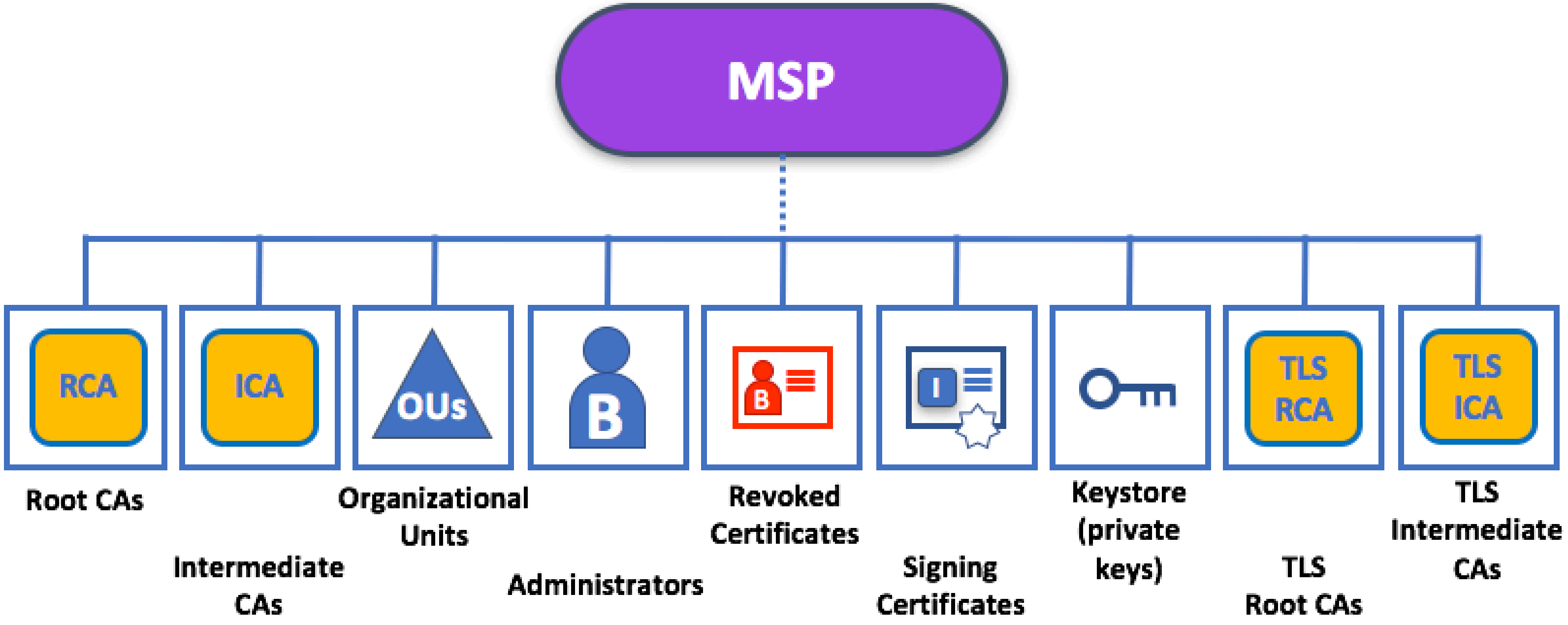
Green certificates signed by ICA1

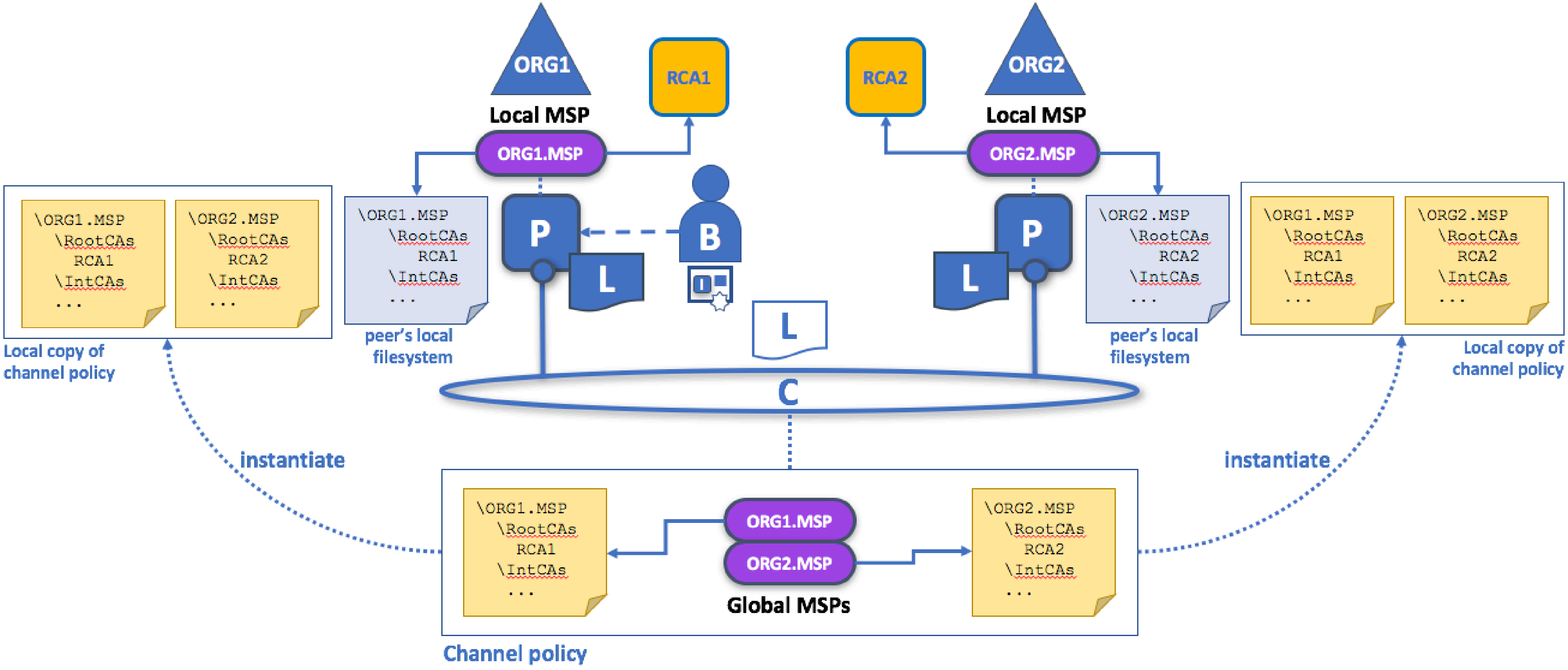
Blue certificates signed by ICA2

Grey certificates signed by ICA3

맴버십 서비스

- it identifies which Root CAs and Intermediate CAs are trusted to define the members of a trust domain
- An MSP can identify specific **roles** an actor might play either within the scope of the organization the MSP represents





ORG1

Local MSP

ORG1.MSP

RCA1

RCA2

ORG2

Local MSP

ORG2.MSP

P

L

B

L

P

L

C

```
\ORG1.MSP
  \RootCAs
    RCA1
  \IntCAs
  ...
```

```
\ORG2.MSP
  \RootCAs
    RCA2
  \IntCAs
  ...
```

Local copy of channel policy

```
\ORG1.MSP
  \RootCAs
    RCA1
  \IntCAs
  ...
```

peer's local filesystem

```
\ORG2.MSP
  \RootCAs
    RCA2
  \IntCAs
  ...
```

peer's local filesystem

```
\ORG1.MSP
  \RootCAs
    RCA1
  \IntCAs
  ...
```

```
\ORG2.MSP
  \RootCAs
    RCA2
  \IntCAs
  ...
```

Local copy of channel policy

instantiate

instantiate

```
\ORG1.MSP
  \RootCAs
    RCA1
  \IntCAs
  ...
```

ORG1.MSP

ORG2.MSP

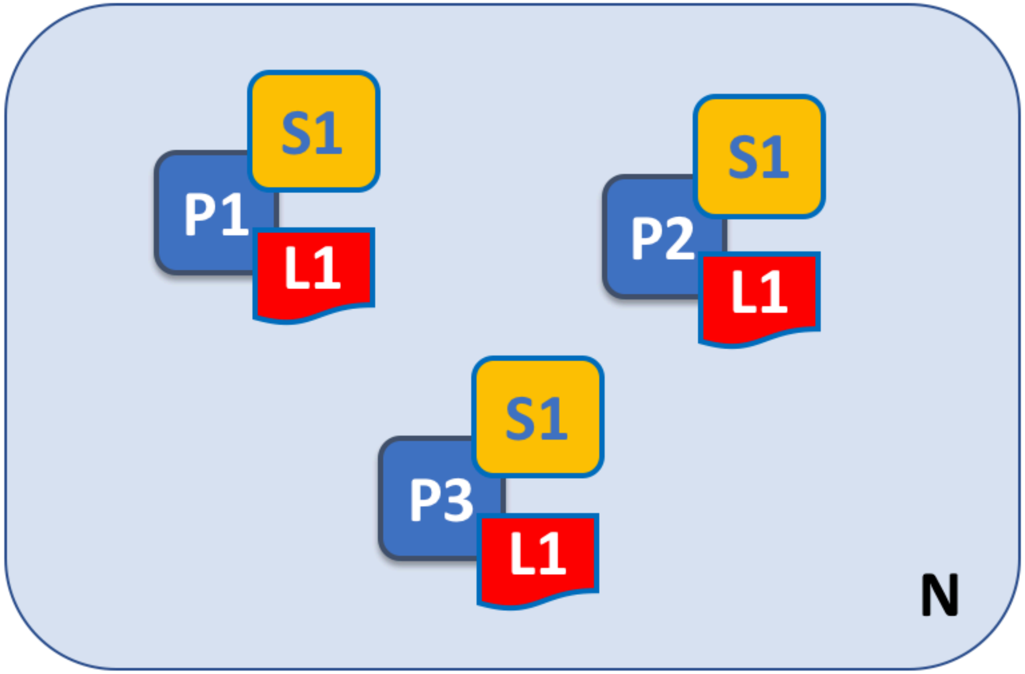
Global MSPs

```
\ORG2.MSP
  \RootCAs
    RCA2
  \IntCAs
  ...
```

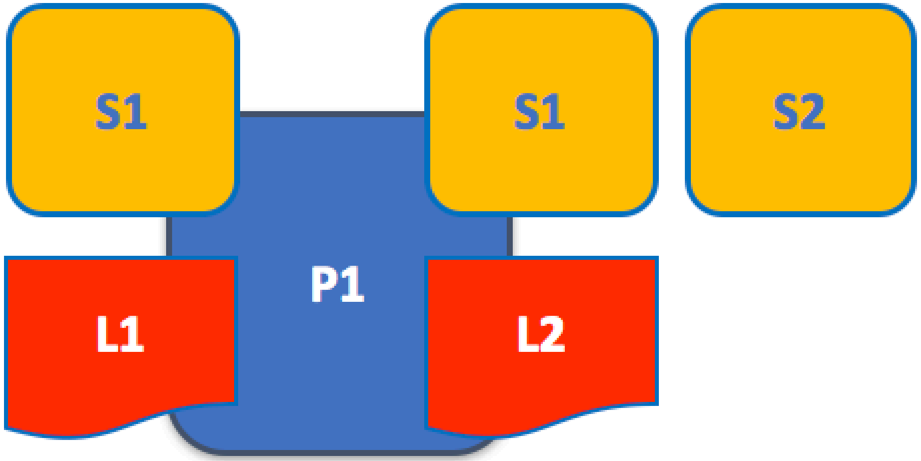
Channel policy

Peers

- Peers are a fundamental element of the network
- host ledgers and smart contracts
- Peers can be created, started, stopped, reconfigured, and even deleted.



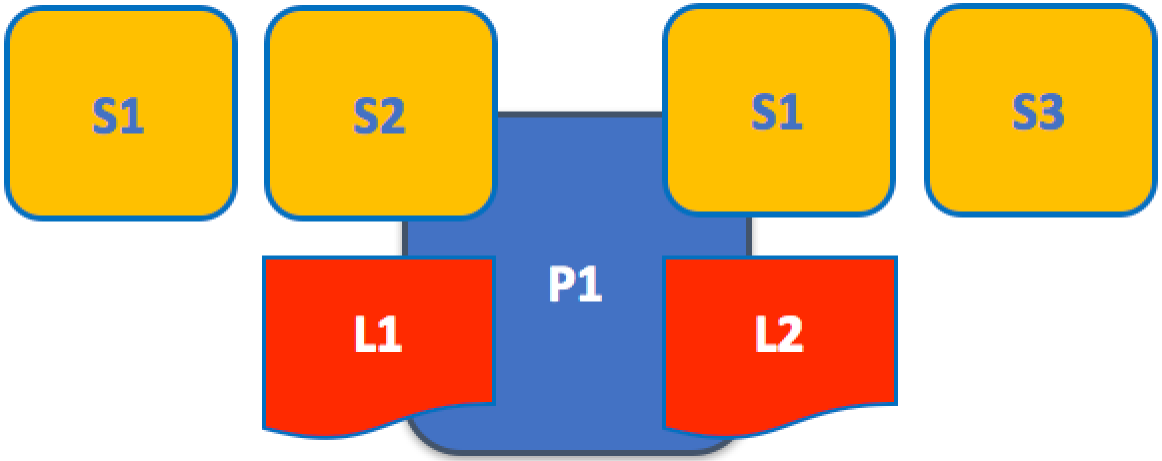
N	Blockchain network
P	Peer node
S	Smart contract (aka chaincode)
L	Ledger



S1 S1 S2

P1

L1 L2



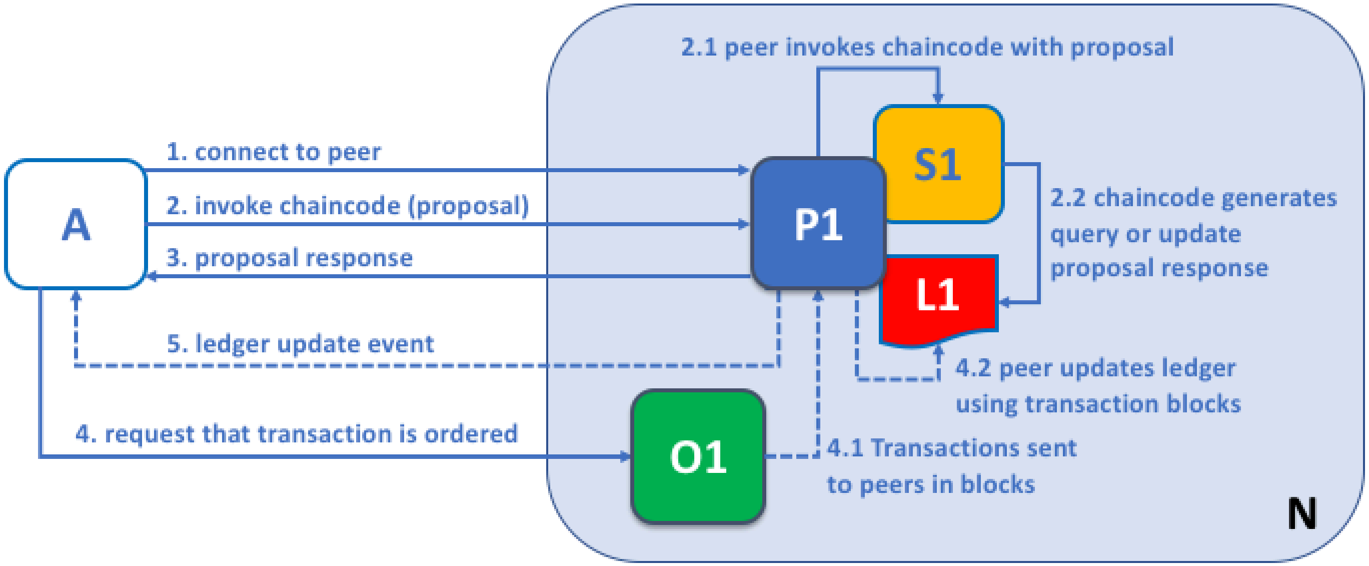
S1 S2 S1 S3

P1

L1 L2

응용프로그램

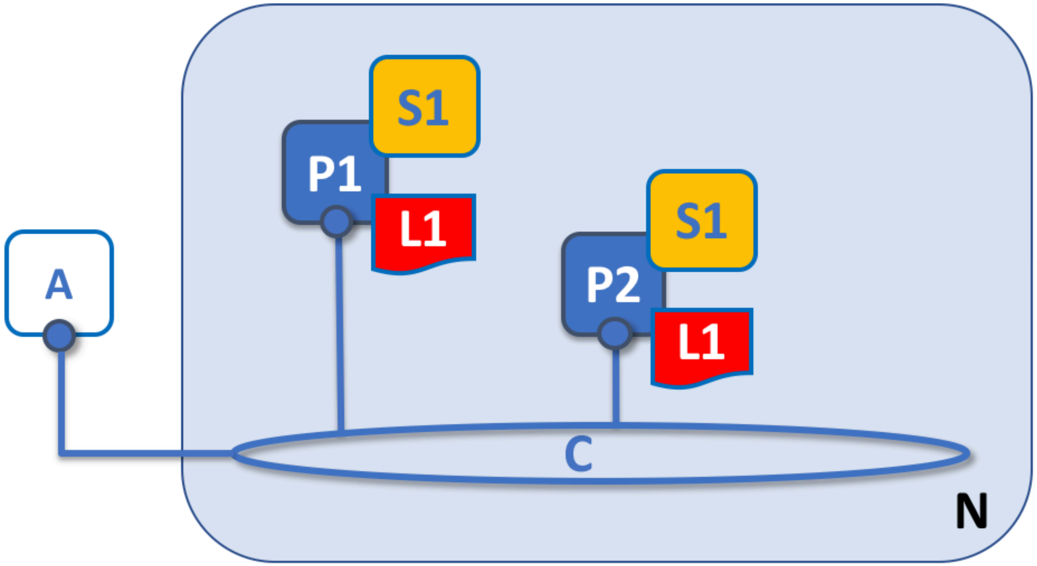
- Fabric Software Development Kit (SDK)


2.1 peer invokes chaincode with proposal
1. connect to peer
2. invoke chaincode (proposal)
3. proposal response
A
P1
S1
L1
2.2 chaincode generates query or update proposal response
5. ledger update event
4.2 peer updates ledger using transaction blocks
4. request that transaction is ordered
O1
4.1 Transactions sent to peers in blocks
N

N	Blockchain Network
A	Application
P	Peer
S	Chaincode
L	Ledger
O	Orderer

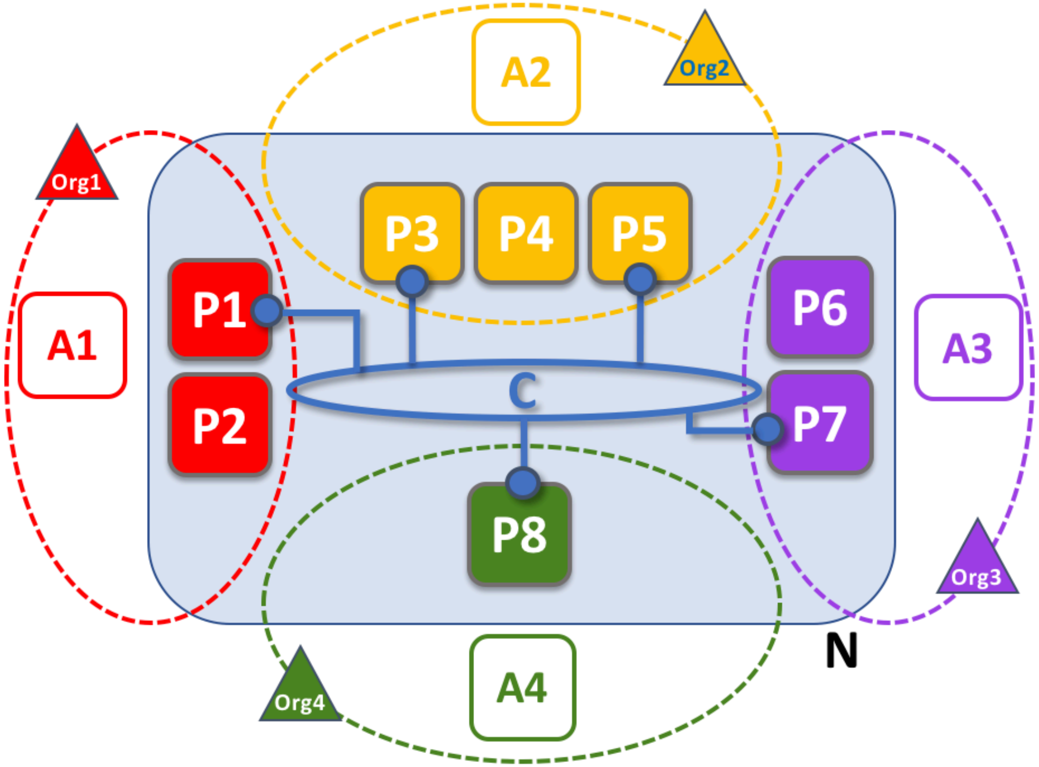
피어와 채널

- *peers provide the control point for access to, and management of, channels.*



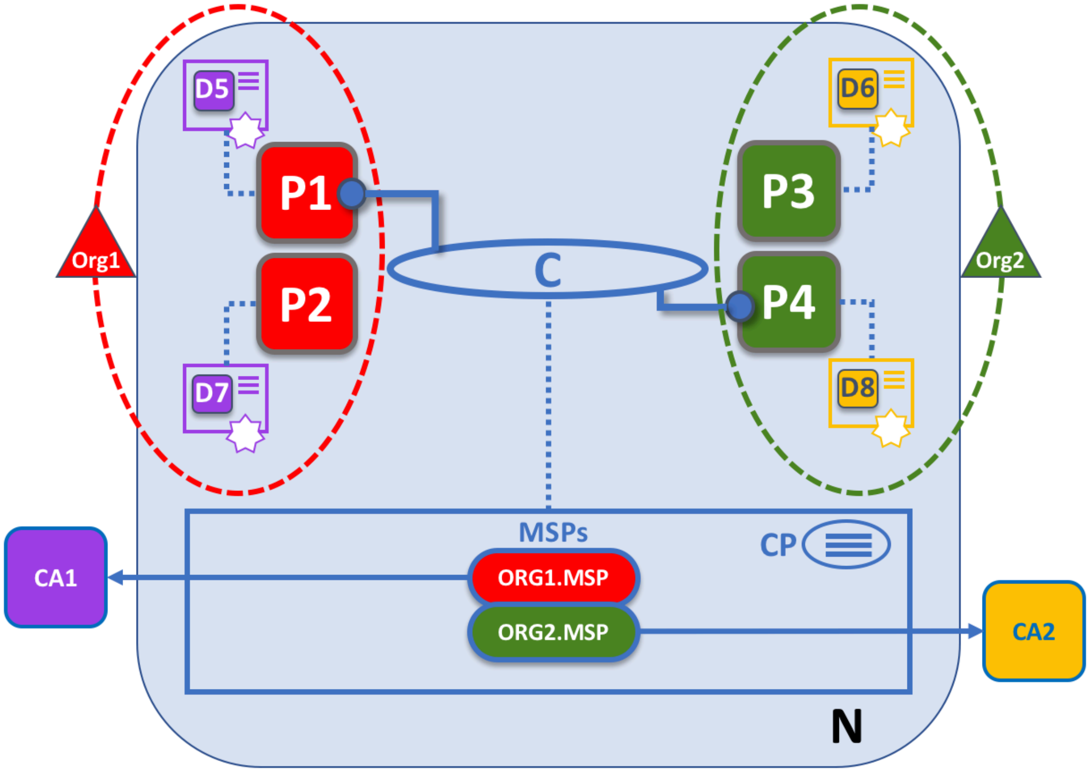
N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer	PA C	Principal PA (e.g. A, P1) communicates via channel C.
S	Chaincode		

피어와 기관



N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer	PA C	Principal PA (e.g. A1, P5) communicates via channel C.
		Org	Organization
A1 R P1 P2		Organization R owns application A1 and peers P1, P2.	

피어와 Identity

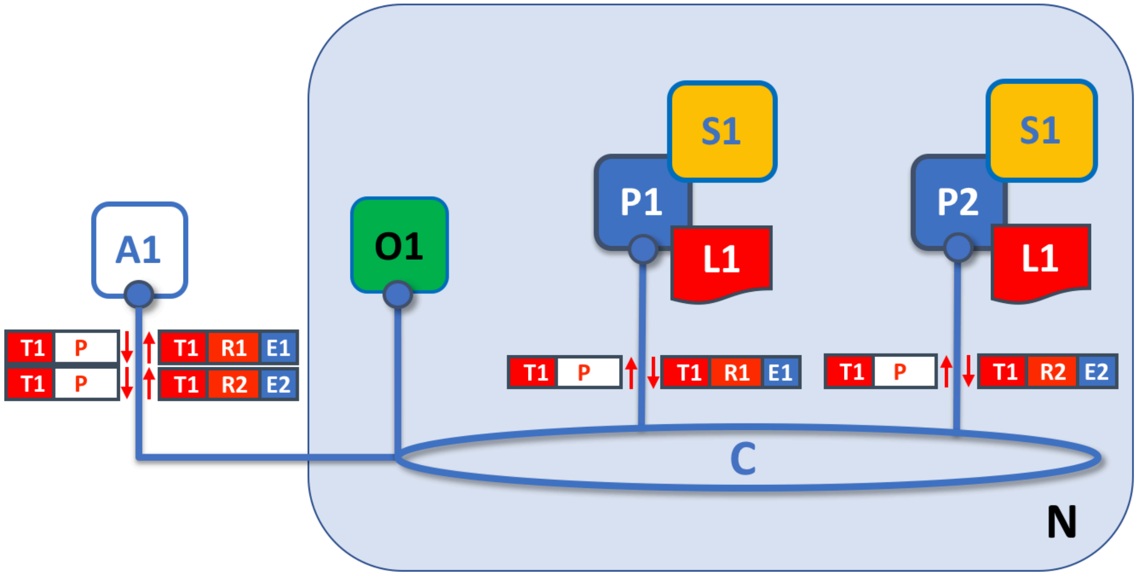


N	Blockchain Network	P	Peer
C	Channel	Org	Organization
I	Identity	PA C	Principal PA (e.g. P1,P4) communicates via channel C.
CP	Channel policy		
CA	Certificate Authority	MSP	Membership Service Provider
A1 R P1 P2		Organization R owns application A1 and peers P1, P2.	
C CP	Channel C subject to policy CP.	MSP1 CP MSP2	Channel policy CP contains MSPs: MSP1 and MSP2.
MSP1 → CA1		MSP1 selects the Certificate Authority CA1 to provide certificates for it.	

D PA	Principal P has identity D

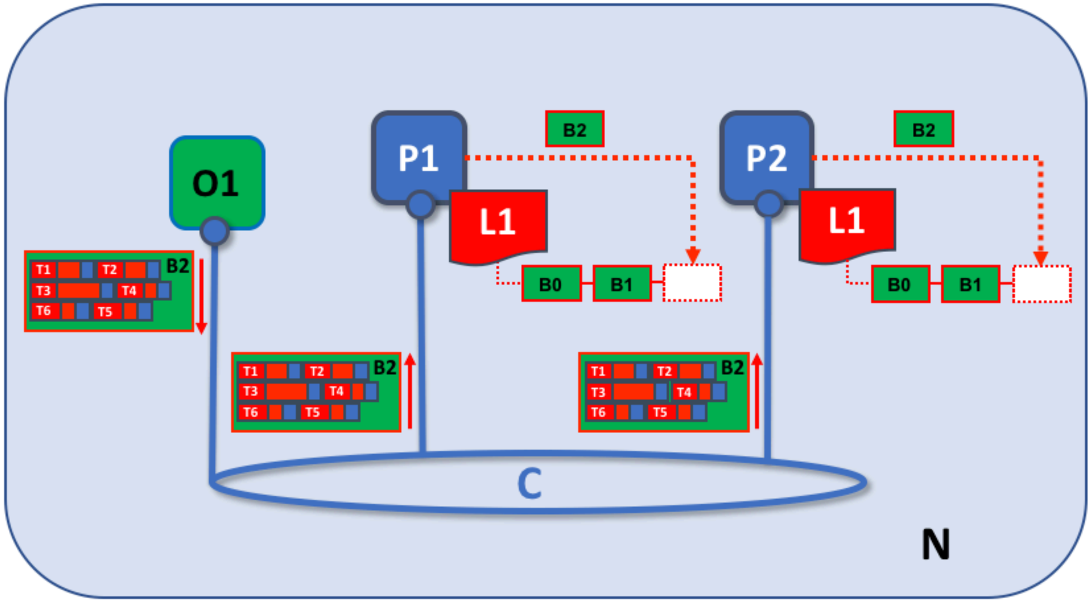
피어와 Orderer

- Phase 1: Proposal



N	Blockchain Network	S	Chaincode
C	Channel	O	Orderer
P	Peer	L	Ledger
T1 P	Transaction T proposal P	T1 R2 E2	Transaction T1, response R2 endorsed with E2
T T C	Ledger transaction T1 flows on channel C	PA C	Principal PA (P1,P2) communicates via channel C.

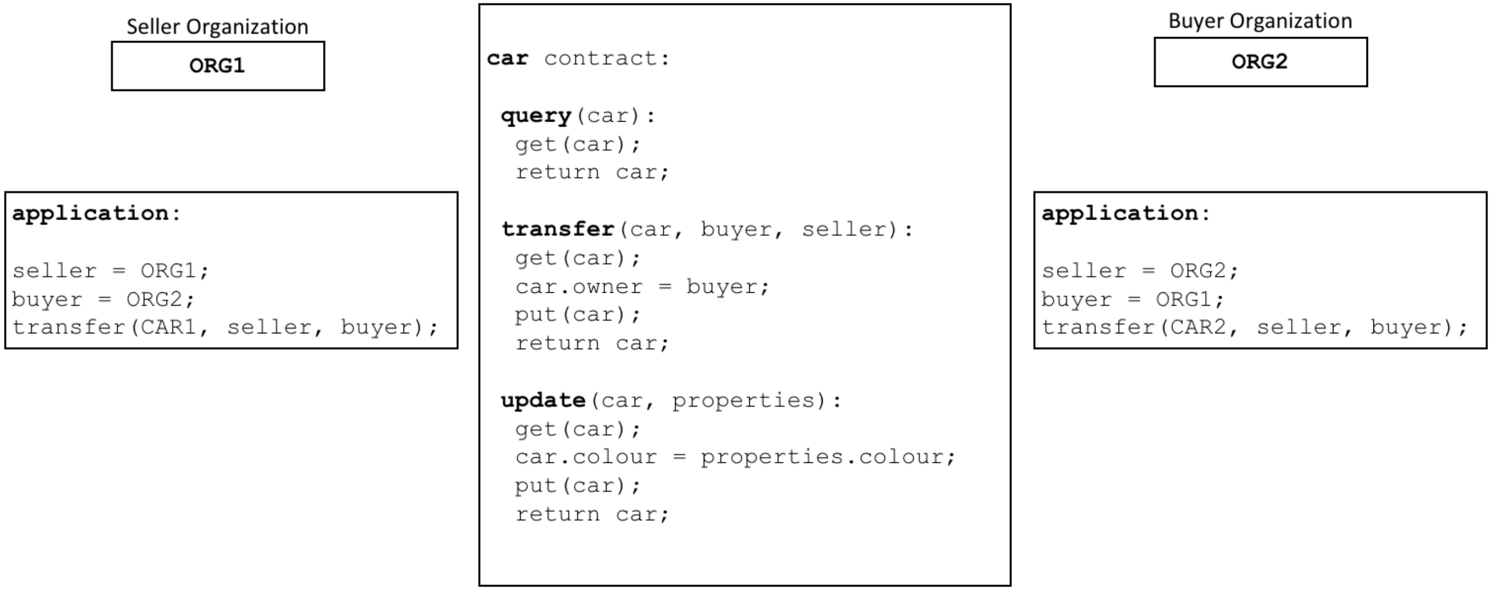
- Phase 2: Ordering and packaging transactions into blocks
- Phase 3: Validation and commit



N	Blockchain Network	P	Peer
C	Channel	O	Orderer
L	Ledger	B	Block B
L1 B0 B1	Ledger L1 has blockchain with blocks B0, B1	T1 T2 T3 B1	Block B1 contains transactions T1, T2, T3...
B1 C	Block B1 flows on channel C	PA C	Principal PA (P1, P2) communicates via channel C.

스마트컨트랙트 & 체인코드

- *A smart contract defines the* ***rules*** *between different organizations in* ***executable code***
- ***Applications invoke a smart contract*** *to generate transactions that are recorded on the ledger.*



Seller Organization

ORG1

```
application:

seller = ORG1;
buyer = ORG2;
transfer(CAR1, seller, buyer);
```

```
car contract:

 query(car):
  get(car);
  return car;

 transfer(car, buyer, seller):
  get(car);
  car.owner = buyer;
  put(car);
  return car;

 update(car, properties):
  get(car);
  car.colour = properties.colour;
  put(car);
  return car;
```

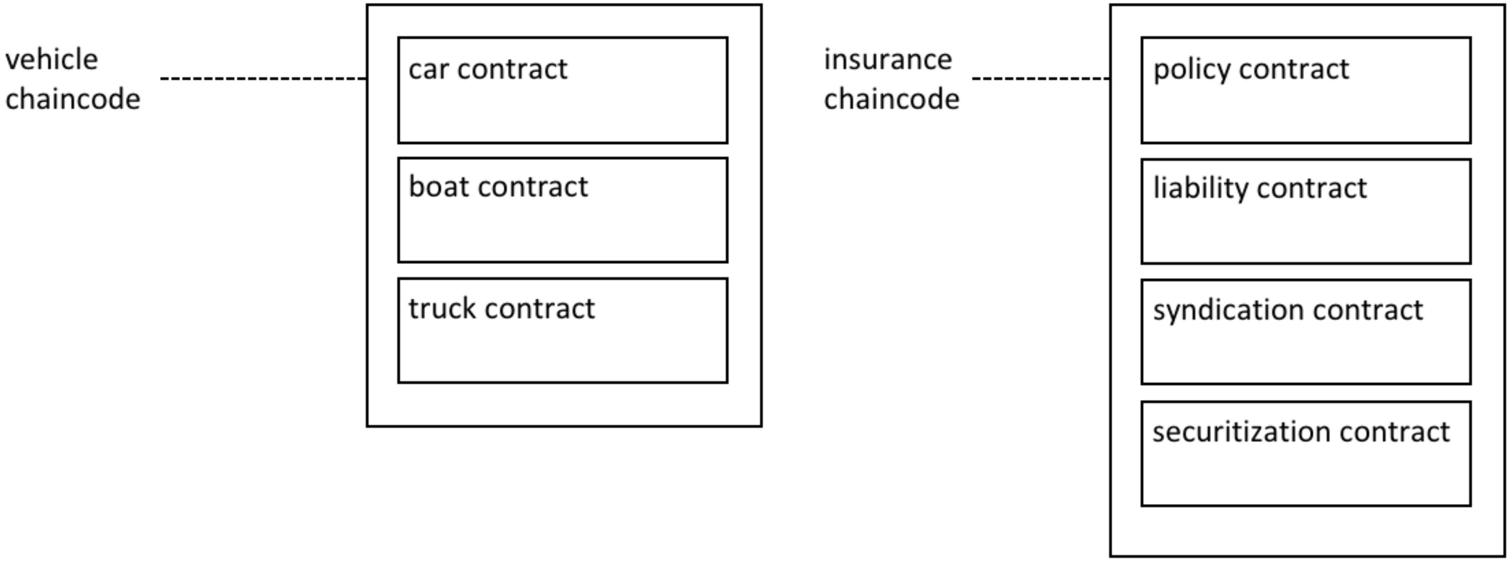
Buyer Organization

ORG2

```
application:

seller = ORG2;
buyer = ORG1;
transfer(CAR2, seller, buyer);
```

용어

- **smart contract**
  defines the **transaction logic**
  that controls the **lifecycle of a business object**
  contained in the **world state.**
- *A* ***smart contract*** *is defined within a* ***chaincode.***



원장 Ledger

- a blockchain immutably records transactions which update states in a ledger.
- **blockchain** – records the history of all transactions
- **world state** – holds a cache of the current value of these states
- Smart contracts
  - **put**, **get** and **delete** states in the world state
  - **query** the immutable blockchain record of transactions

get

represents a query to retrieve information about the current state of a business object

put

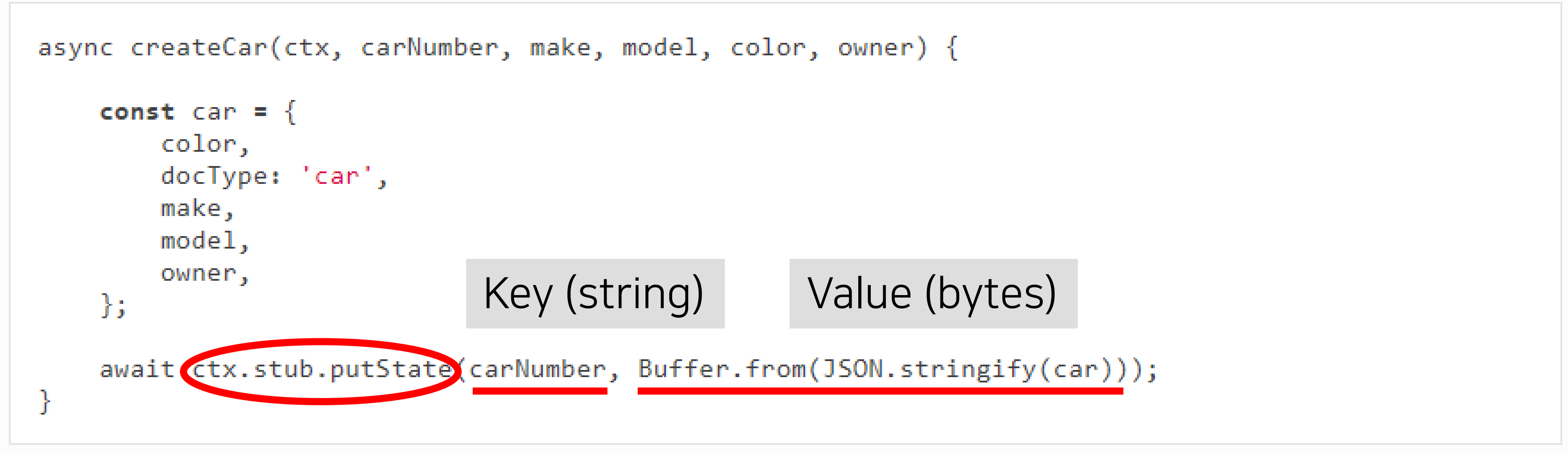
creates a new business object or modifies an existing one in the ledger world state

delete

represents removal of a business object from the current state of the ledger, but not its history

개발

- Fabcar smart contract in the "Writing your first application" tutorial.
- JavaScript, GOLANG or Java.



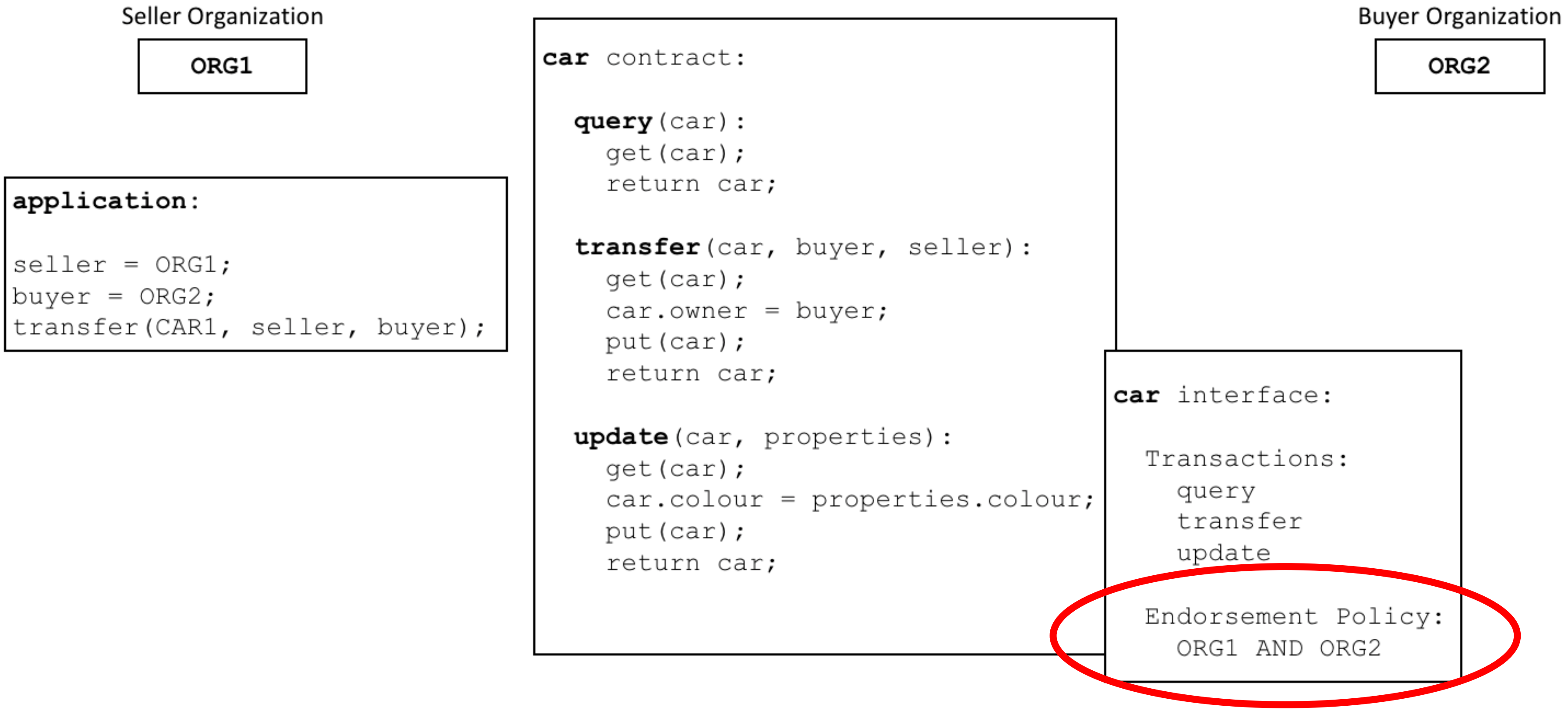
```
async createCar(ctx, carNumber, make, model, color, owner) {

    const car = {
        color,
        docType: 'car',
        make,
        model,
        owner,
    };

    await ctx.stub.putState(carNumber, Buffer.from(JSON.stringify(car)));
}
```

Key (string)

Value (bytes)

Endorsement

- *Every smart contract has an endorsement policy*



Seller Organization

ORG1

```
application:

seller = ORG1;
buyer = ORG2;
transfer(CAR1, seller, buyer);
```

```
car contract:

  query(car):
    get(car);
    return car;

  transfer(car, buyer, seller):
    get(car);
    car.owner = buyer;
    put(car);
    return car;

  update(car, properties):
    get(car);
    car.colour = properties.colour;
    put(car);
    return car;
```

Buyer Organization

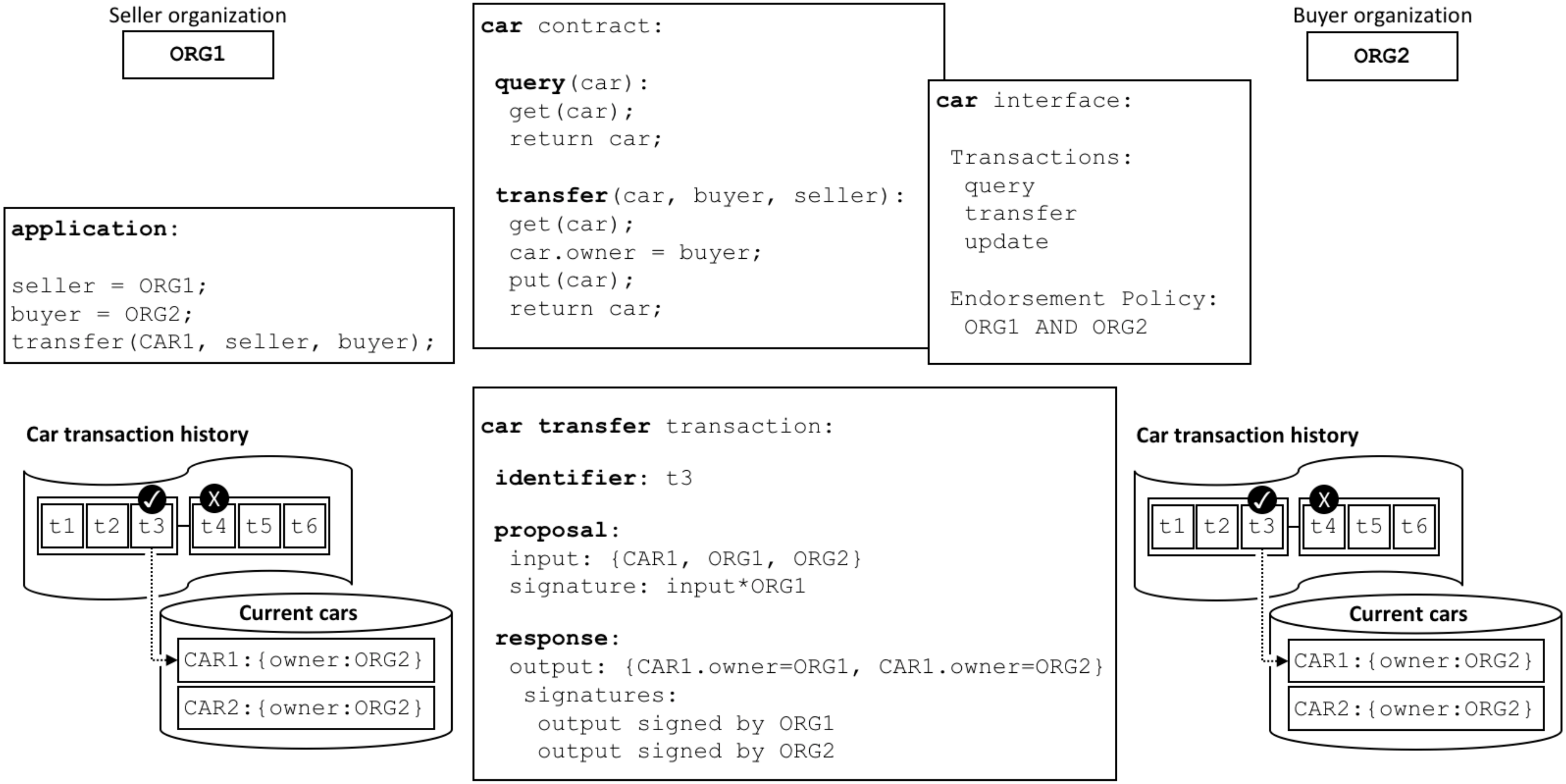
ORG2

```
car interface:

  Transactions:
    query
    transfer
    update

  Endorsement Policy:
    ORG1 AND ORG2
```

When? Instantiate, Upgrade

유효한 거래

- When a smart contract executes, it **runs on a peer node** owned by an organization
- **transaction proposal**
- **transaction proposal response** (with **read-write set**)
- both the states that have been read, and the new states that are to be written if the transaction is valid. Notice that the world state **is not updated when the smart contract is executed**!


Seller organization

ORG1

Buyer organization

ORG2

```
application:

seller = ORG1;
buyer = ORG2;
transfer(CAR1, seller, buyer);
```

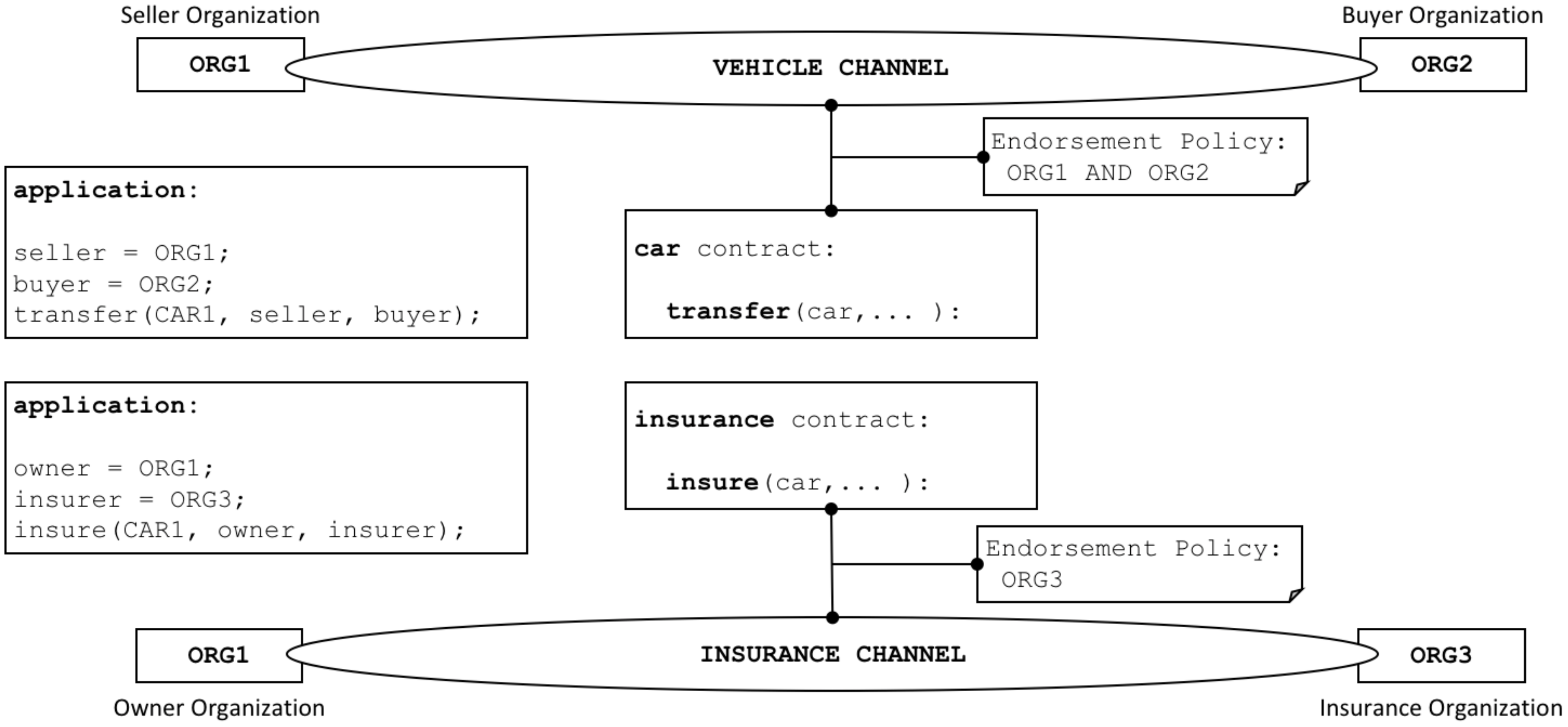
```
car contract:

 query(car):
  get(car);
  return car;

 transfer(car, buyer, seller):
  get(car);
  car.owner = buyer;
  put(car);
  return car;
```

```
car interface:

 Transactions:
  query
  transfer
  update

 Endorsement Policy:
  ORG1 AND ORG2
```

```
car transfer transaction:

 identifier: t3

 proposal:
  input: {CAR1, ORG1, ORG2}
  signature: input*ORG1

 response:
  output: {CAR1.owner=ORG1, CAR1.owner=ORG2}
   signatures:
    output signed by ORG1
    output signed by ORG2
```

채널

- network of networks
- communications privacy



```
application:

seller = ORG1;
buyer = ORG2;
transfer(CAR1, seller, buyer);
```

```
car contract:

  transfer(car,... ):
```

```
application:

owner = ORG1;
insurer = ORG3;
insure(CAR1, owner, insurer);
```

```
insurance contract:

  insure(car,... ):
```

시스템 체인코드

- define low-level program code which corresponds to domain independent **system** interactions

Lifecycle system chaincode (LSCC)

to handle package signing, install, instantiate, and upgrade chaincode requests

Configuration system chaincode (CSCC)

to handle changes to a channel configuration, such as a policy update

Query system chaincode (QSCC)

to provide ledger APIs which include block query, transaction query etc.

Endorsement system chaincode (ESCC)

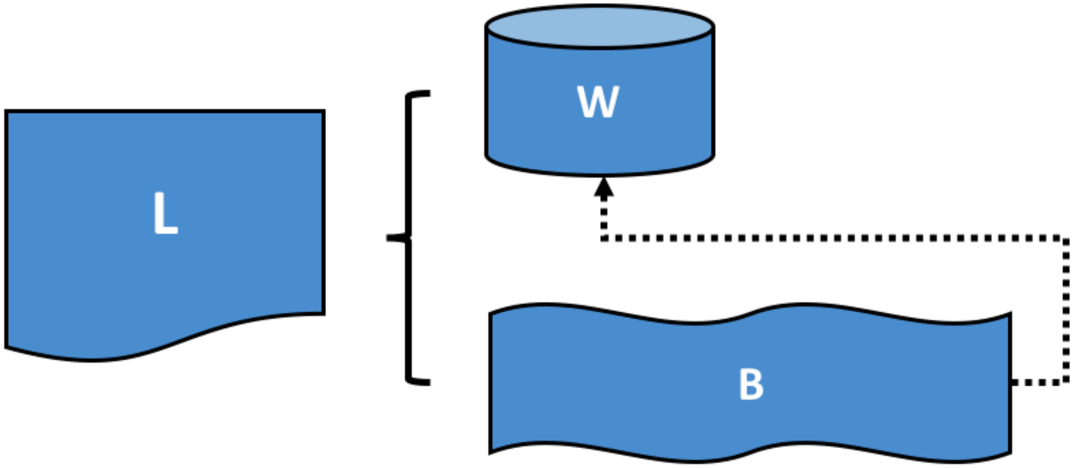
to cryptographically sign a transaction response.

Validation system chaincode (VSCC)

validates a transaction, including checking endorsement policy and read-write set versioning.

원장 – Ledger

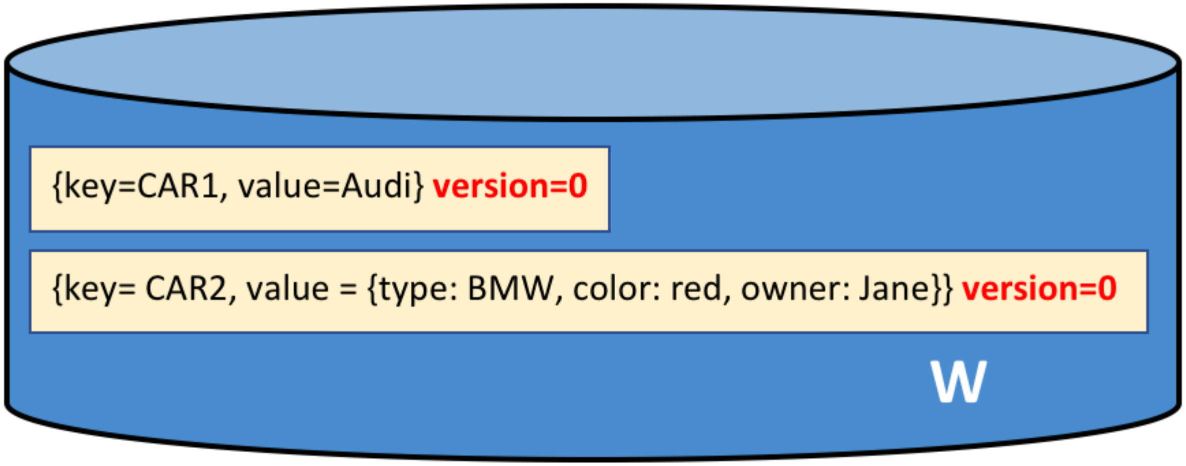
- *A Ledger L comprises blockchain B and world state W, where* ***blockchain B determines world state W.***
- *We can also say that world state W is derived from blockchain B.*



L	Ledger
W	World State
B	Blockchain
L { B W	L comprises B and W
W ◄┐ B ┘	B determines W

World State

- useful because programs usually require the current value of an object
- simple ledger APIs to **get**, **put** and **delete** states.
- implemented as a database



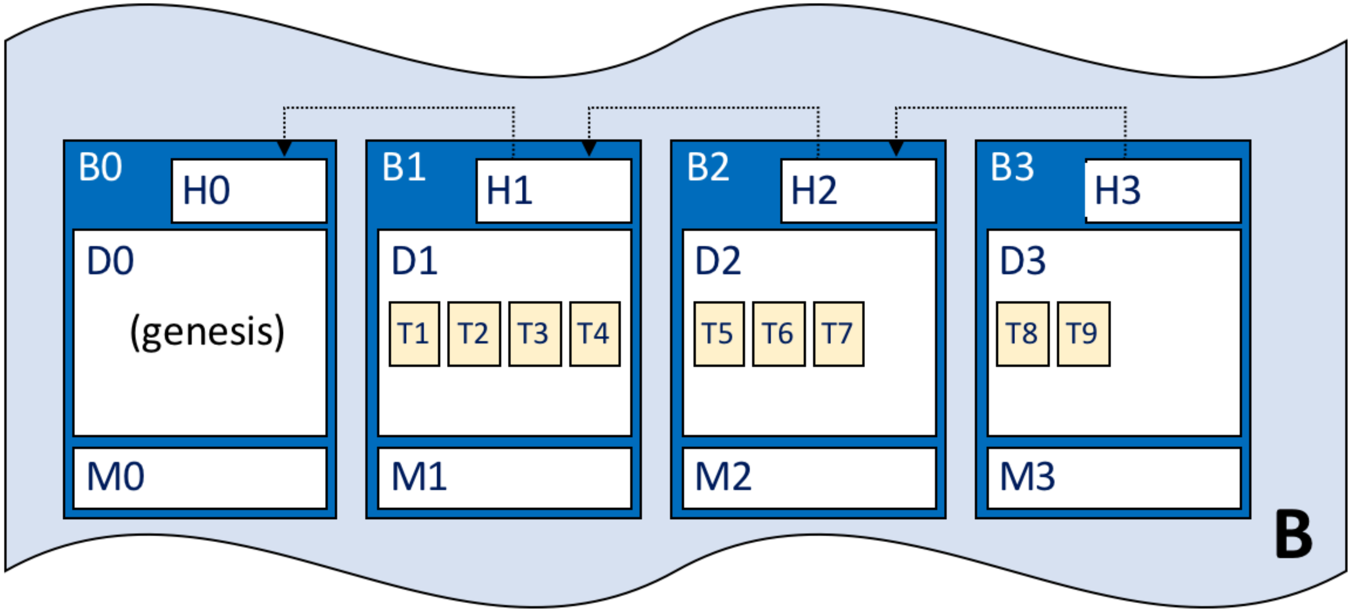
{key=CAR1, value=Audi} **version=0**

{key= CAR2, value = {type: BMW, color: red, owner: Jane}} **version=0**

W	Ledger world state
{key=**K**, value = **V** } **version=0**	A ledger state with **key=K**. It contains a set of facts expressed as a simple value, **V**. The state is at version 0.
{key=**K**, value = {**KV**} } **version=0**	A ledger state with **key=K**. It contains a set of facts expressed as a set of key-value pairs **{KV}.** The state is at version 0.

Blockchain

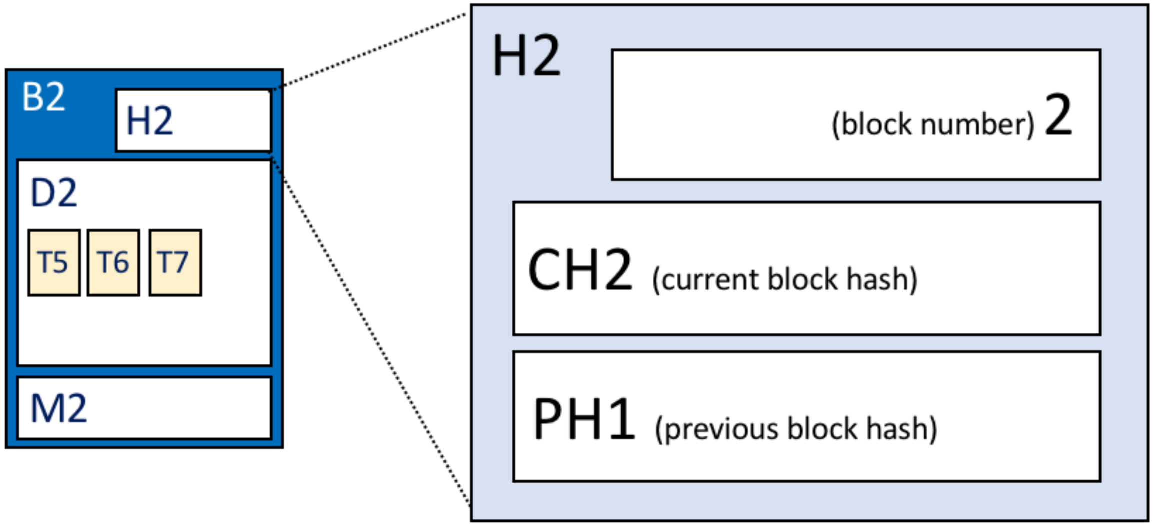
- blockchain has recorded every previous version of each ledger state and how it has been changed.
- blocks are first created by a Hyperledger Fabric component called the **ordering service**
- always implemented as a file



B	Blockchain
B1	Block
H3	Block header
D1	Block data
T5	Transaction
M3	Block metadata
H1 H2	H2 is chained to H1

Blocks

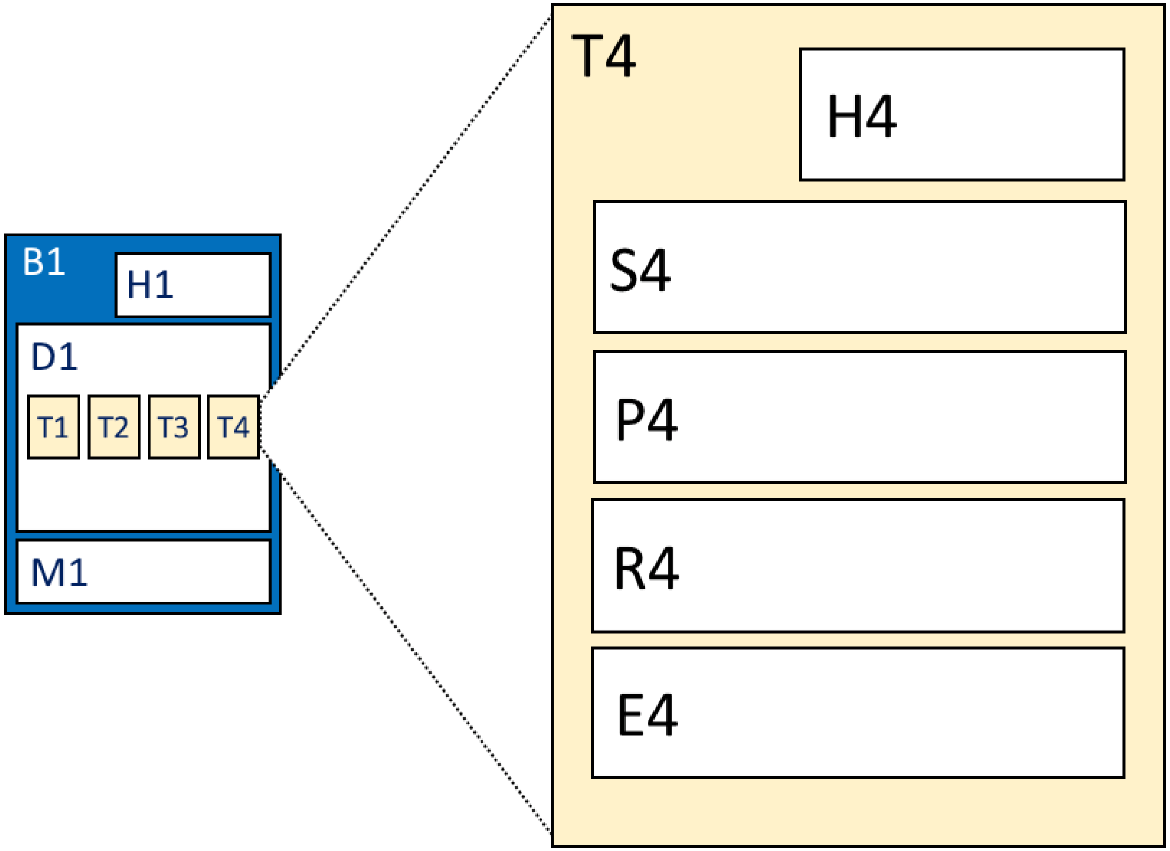
- Block Data
  - a list of transactions arranged in order
- Block Metadata
  - creation time, certificate, public key and signature of block writer, valid/invalid indicator for every transaction



H2	Block header
2	Block number
CH2	Hash of current block transactions
PH1	Copy of hash from previous block
H2 ◁ V2	V2 is detailed view of H2

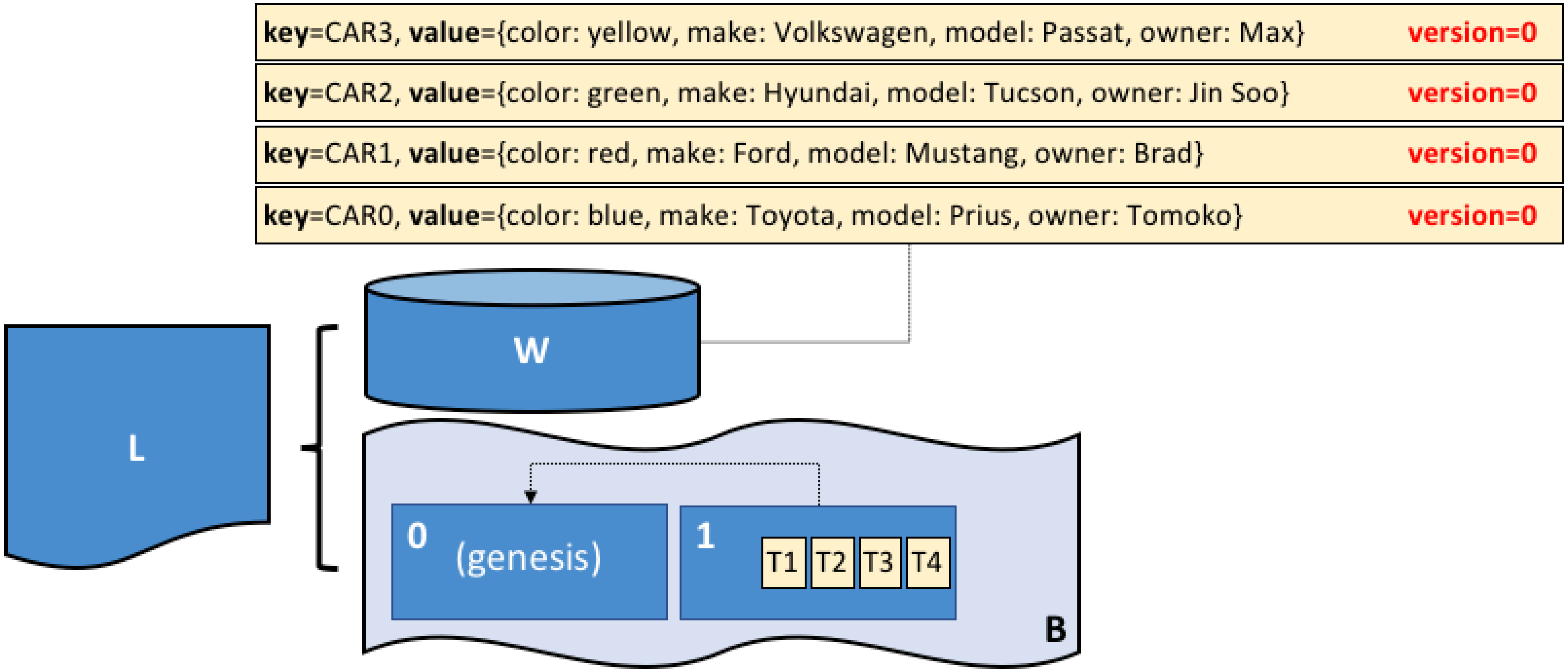
Transactions

- **Header – name of chaincode, version**
- Signature – by client application



T4	Transaction
H4	Header
S4	Signature
P4	Proposal
R4	Response
E4	Endorsements
T4 ◁ V4	V4 is detailed view of T4

Example Ledger: fabcar



key=CAR3, **value**={color: yellow, make: Volkswagen, model: Passat, owner: Max} **version=0**

key=CAR2, **value**={color: green, make: Hyundai, model: Tucson, owner: Jin Soo} **version=0**

key=CAR1, **value**={color: red, make: Ford, model: Mustang, owner: Brad} **version=0**

key=CAR0, **value**={color: blue, make: Toyota, model: Prius, owner: Tomoko} **version=0**

ordering 이란?

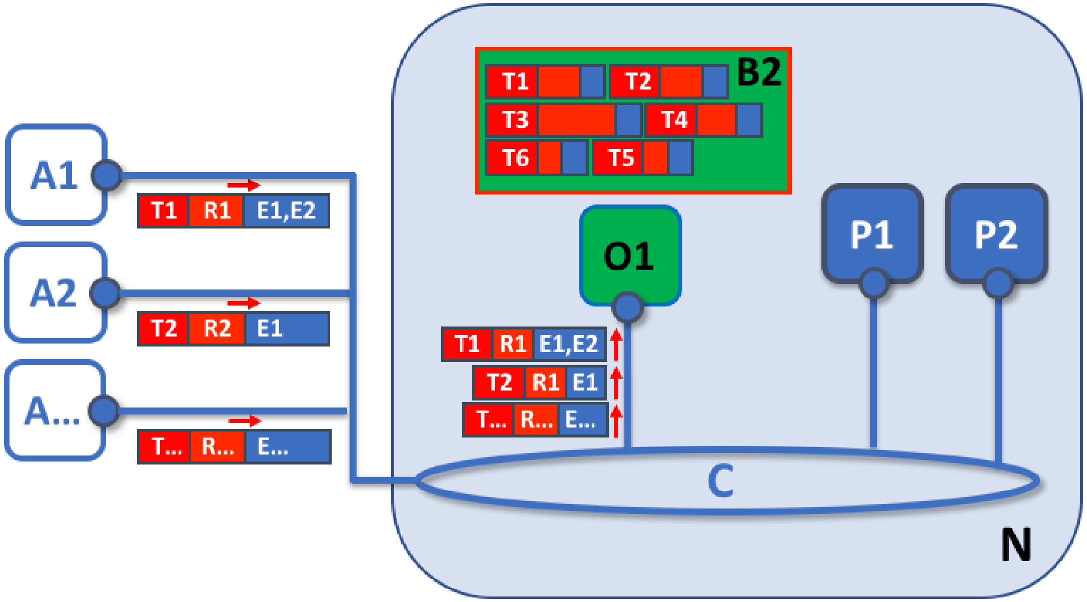
- **deterministic** consensus algorithms
- separating the endorsement of chaincode execution **from ordering**
- basic access control for channels
- Just like peers, ordering nodes belong to an organization including CA(acts as root CA)

거래 흐름 페이즈 1:

- Proposal
- a client application sends a transaction proposal
- subset of peers invoke a smart contract **to produce a proposed ledger update** and then **endorse the results.**
- The endorsing peers **do not apply the proposed update** to their copy of the ledger at this time.
- endorsing peers return a **proposal response** to the client application

거래 흐름 페이즈 2:

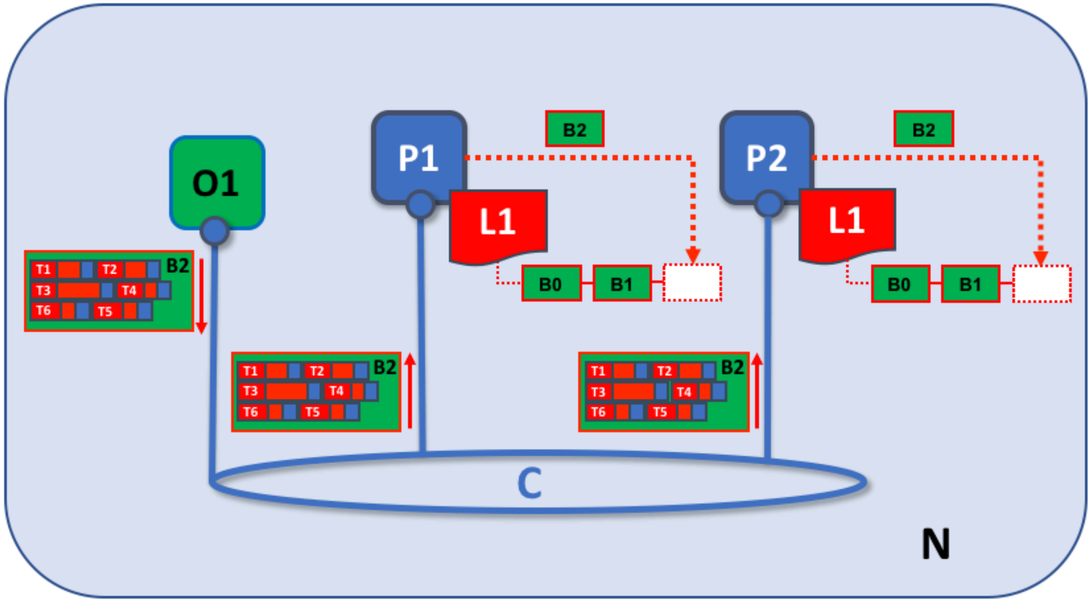
- **Ordering and packaging transactions into blocks**
- application clients submit transactions containing endorsed transaction proposal responses to an ordering service node
- ordering service creates blocks of transactions
- arrange batches of submitted transactions into a well-defined sequence and package them into *blocks*.
- block depends on channel configuration parameters related to the desired size and maximum elapsed duration for a block (BatchSize and BatchTimeout)
- Fabric's finality means that there are no **ledger forks**



N	Blockchain Network	P	Peer
B1	Block B1	O	Orderer
T1 R2a E2	Transaction T1, response R2a endorsed with E2	C	Channel
T1 T2 T3 B1	Block B1 contains transactions T1, T2, T3…		
T1 T1 C	Ledger transaction T1 flows on channel C	PA C	Principal PA (P1,P2) communicates via channel C.

거래 흐름 페이즈 3:

- Validation and commit
- the distribution and subsequent validation of blocks
- cascade blocks to other peers using the gossip protocol



N	Blockchain Network	P	Peer
C	Channel	O	Orderer
L	Ledger	B	Block B
L1 B0 B1	Ledger L1 has blockchain with blocks B0, B1	T1 B1 T2 T3	Block B1 contains transactions T1, T2, T3...
B1 C	Block B1 flows on channel C	PA C	Principal PA (P1, P2) communicates via channel C.

Ordering 서비스 구현

Solo

. a single ordering node

it is not, and never will be, fault tolerant

. a good choice for testing applications and smart contracts, or for creating proofs of concept.

Raft

. crash fault tolerant (CFT)

. a “leader and follower” model

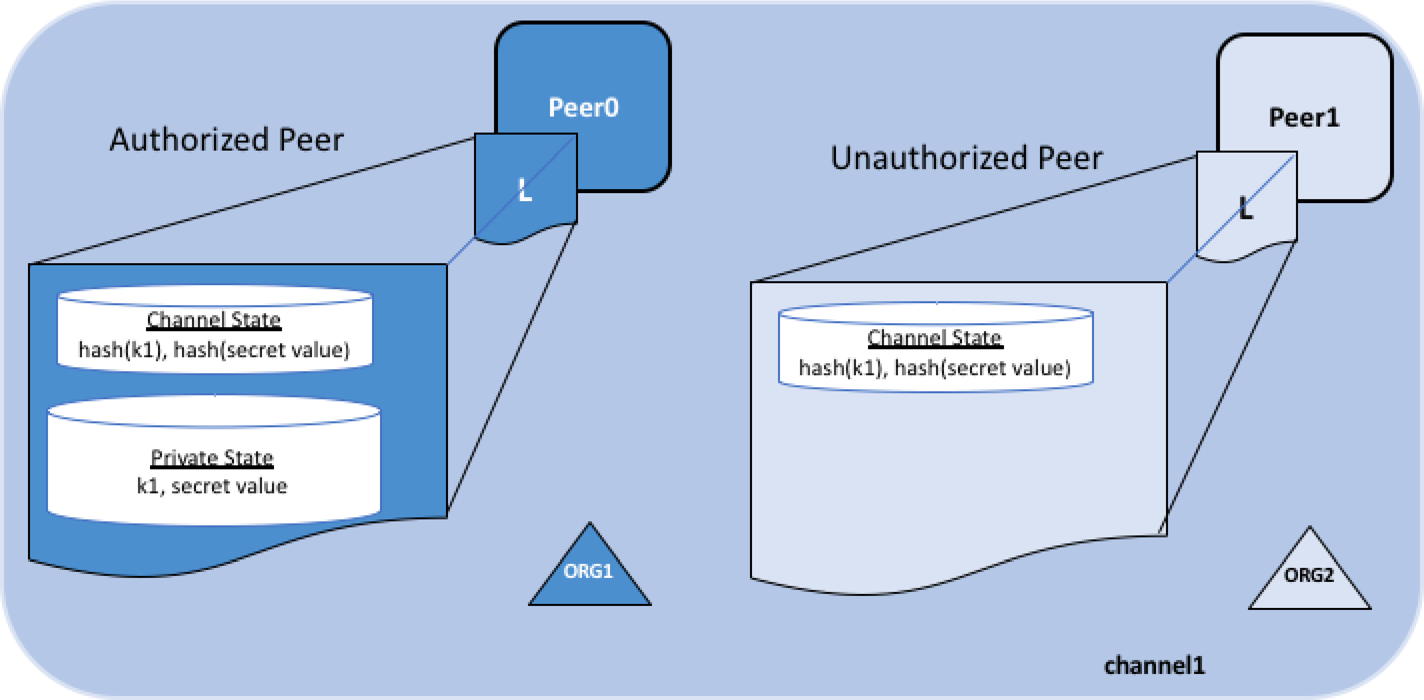
kafka

. CFT implementation that uses a “leader and follower” node configuration

.ZooKeeper ensemble for management purposes

Private Data

- to keep data private from other organizations on that channel

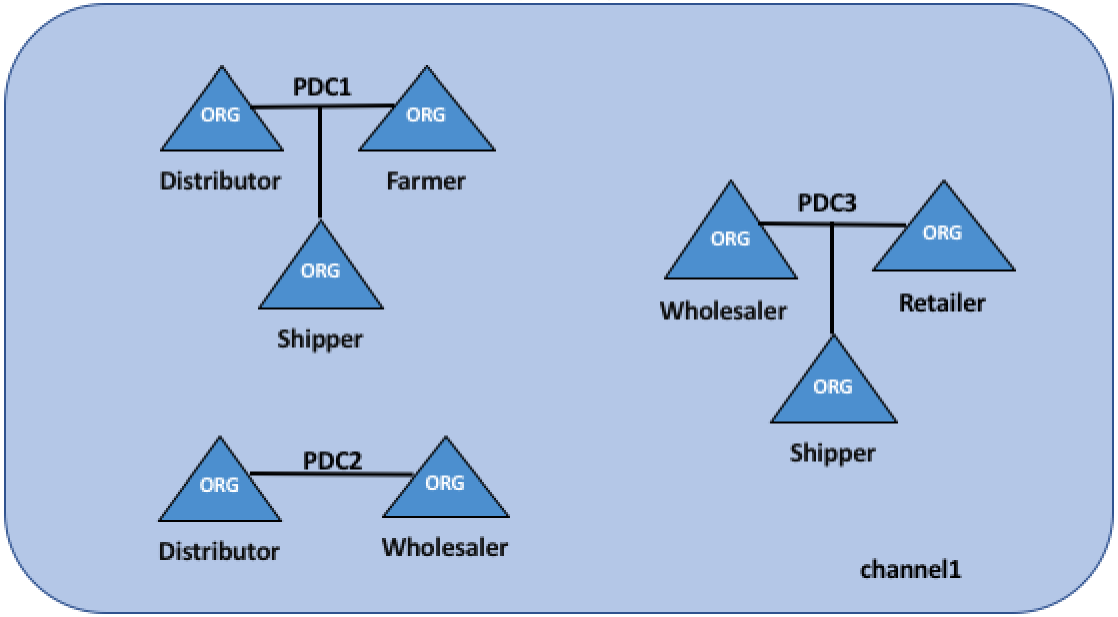


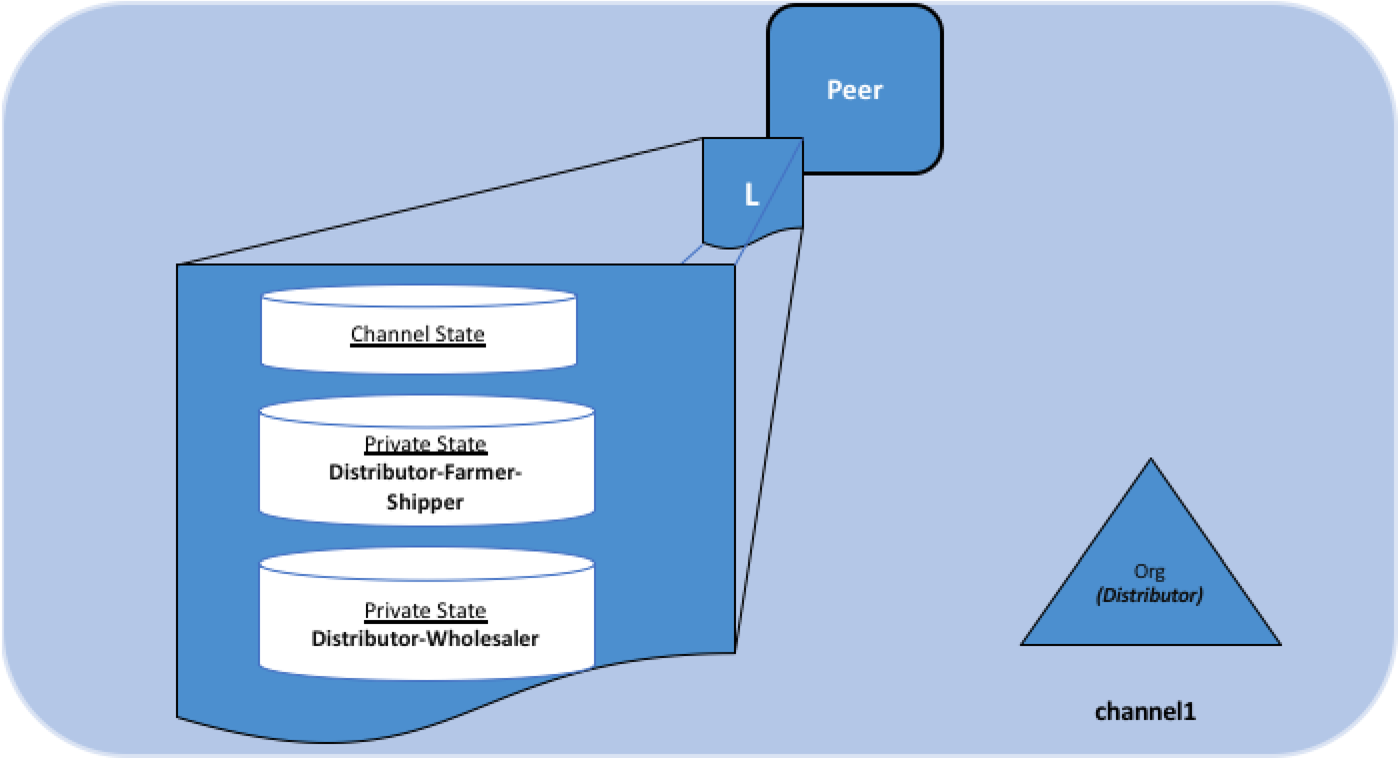
Use case

- A **Farmer** selling his goods abroad
- A **Distributor** moving goods abroad
- A **Shipper** moving goods between parties
- A **Wholesaler** purchasing goods from distributors
- A **Retailer** purchasing goods from shippers and wholesalers

- PDC1: **Distributor, Farmer** and **Shipper**
- PDC2: **Distributor** and **Wholesaler**
- PDC3: **Wholesaler, Retailer** and **Shipper**

Private data collections (PDC)




Peer
L
Channel State
Private State
Distributor-Farmer-Shipper
Private State
Distributor-Wholesaler
Org
(Distributor)
channel1

Thank you.