ApiConf

edoardoconte

BLOCKCHAIN
Technology&Applications

#apiconf2018

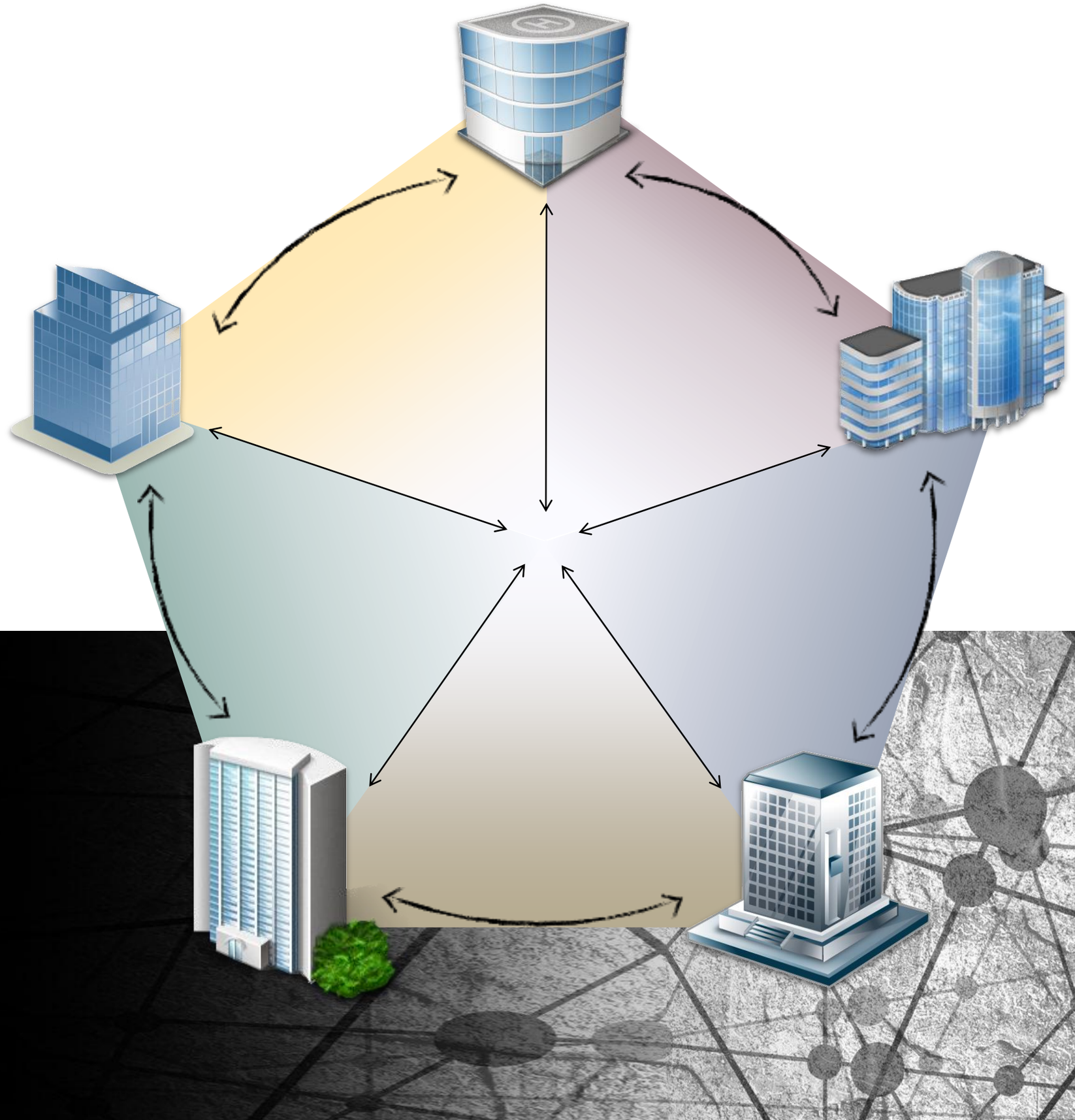
ApiConf

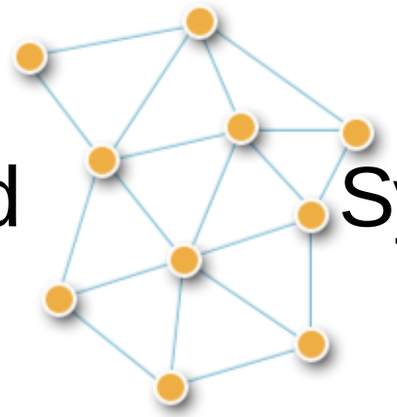
edoardoconte

BLOCKCHAIN
Technology&Applications

#apiconf2018

The Blockchain



Distributed  System

Normally, also  **Decentralized**

Distributed and decentralized

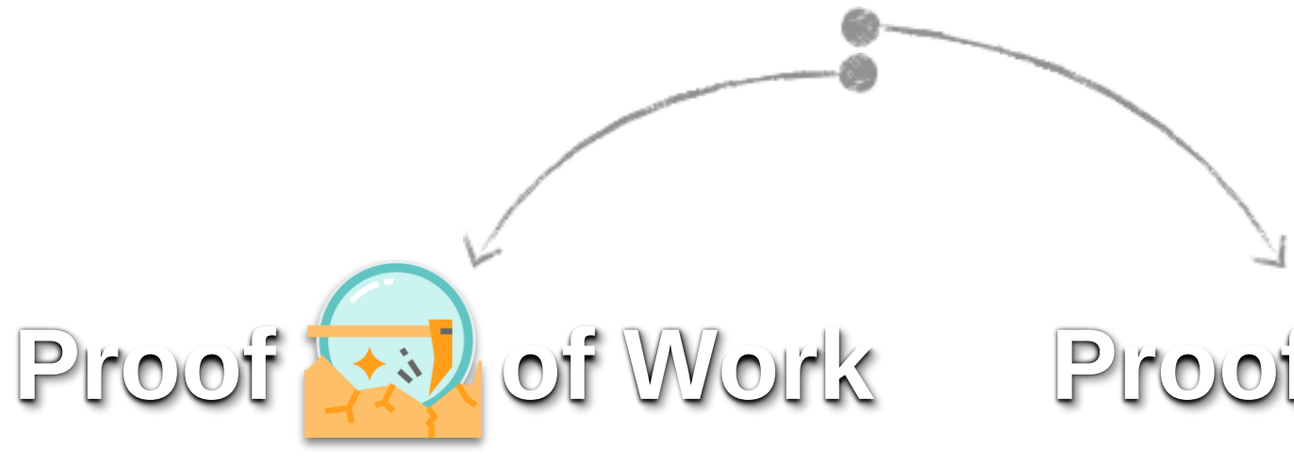
- ***Everyone*** **can join and maintain the blockchain**
- **Possibility to design and deploy *Smart Contracts***
- ***Decentralized*** **and distributed systems**

ethereum IOTA ripple

Public Blockchains

Determine the current builder

Necessary to maintain a unique sequence of blocks!



Consensus Algorithm

Proof of Work

Non-scalable algorithm

Great computational cost

Complexity really low

Proof of Stake

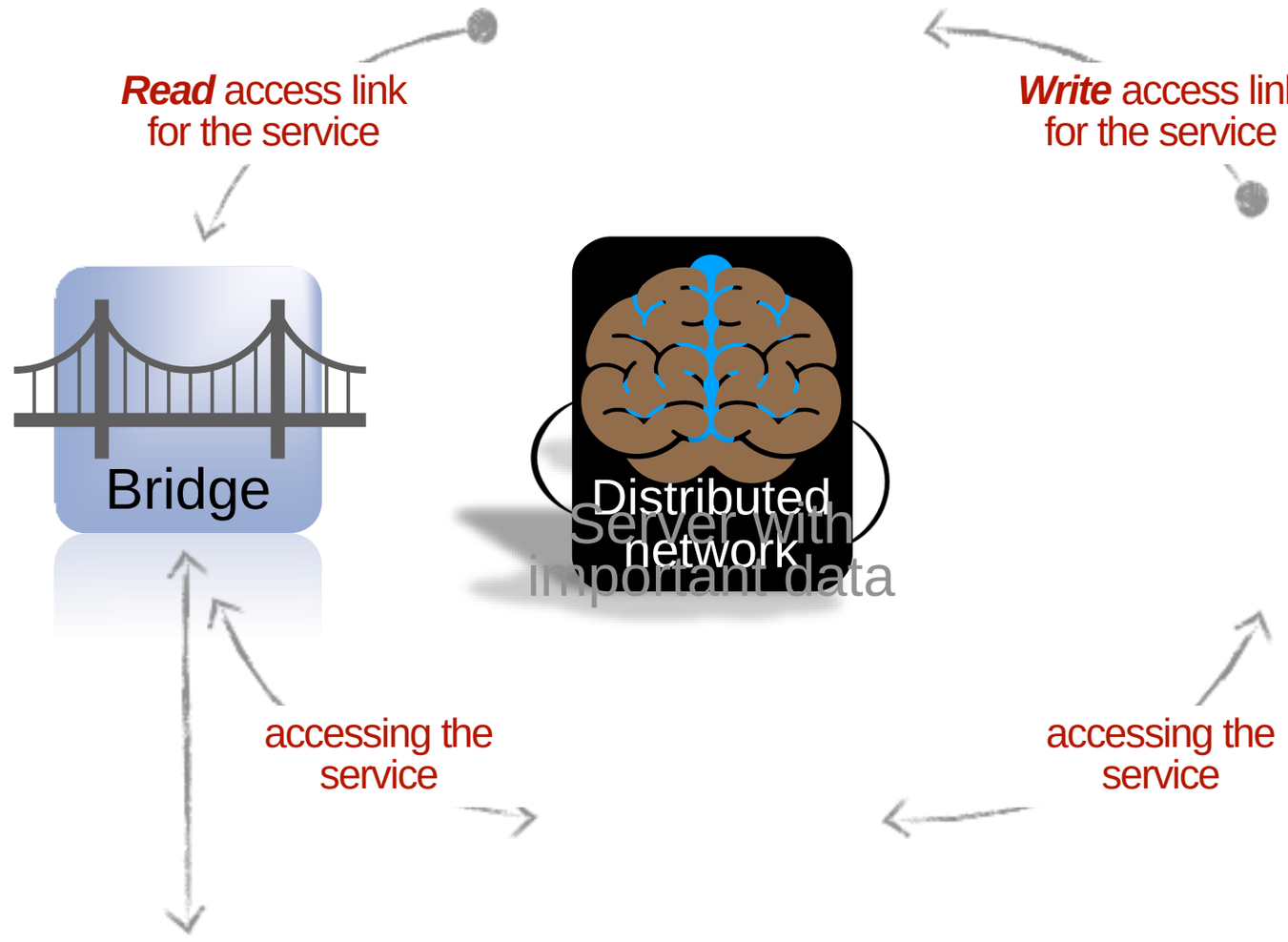
Possibility to scale

Costless algorithm

Very complex to implement

Consensus Algorithm

Secure data through Blockchains



- ***Not*** **completely** ***decentralized***
- **Different authorized entities work together to maintain a common service**
- ***Standardized*** **encryption and replication** ***protocols***

MultiChain HYPERLEDGER BIGCHAINDB

Permissioned Blockchains

a **Full replication of data**

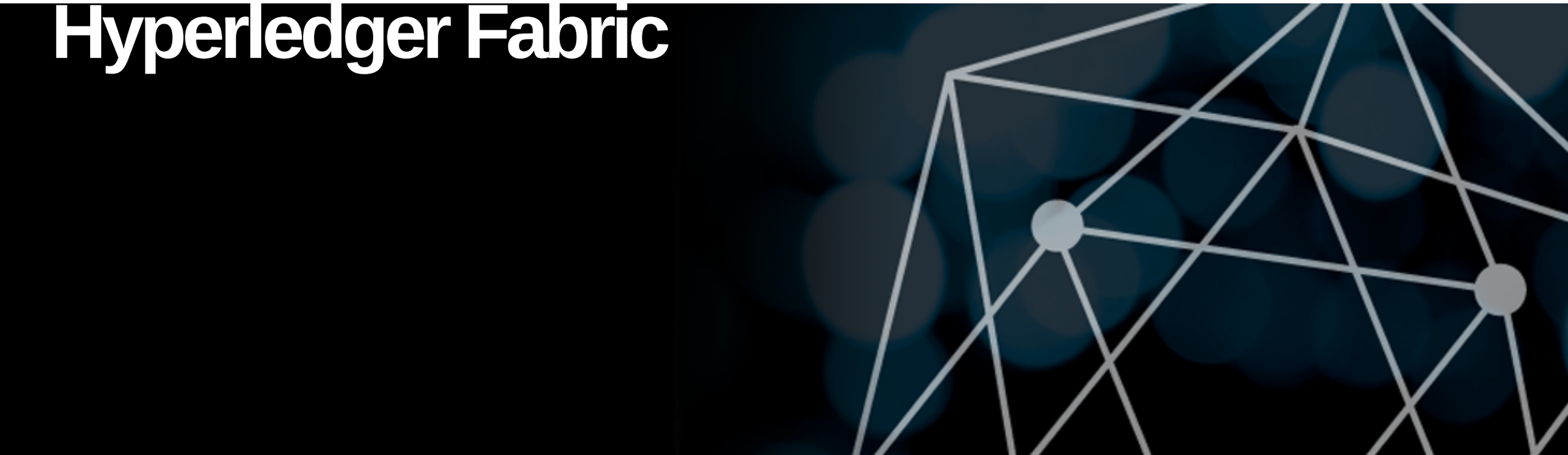
b **Chaincode (Smart Contracts)**

c **Customizable consensus algorithm**

- Crash Fault Tolerant
- Byzantine Fault Tolerant

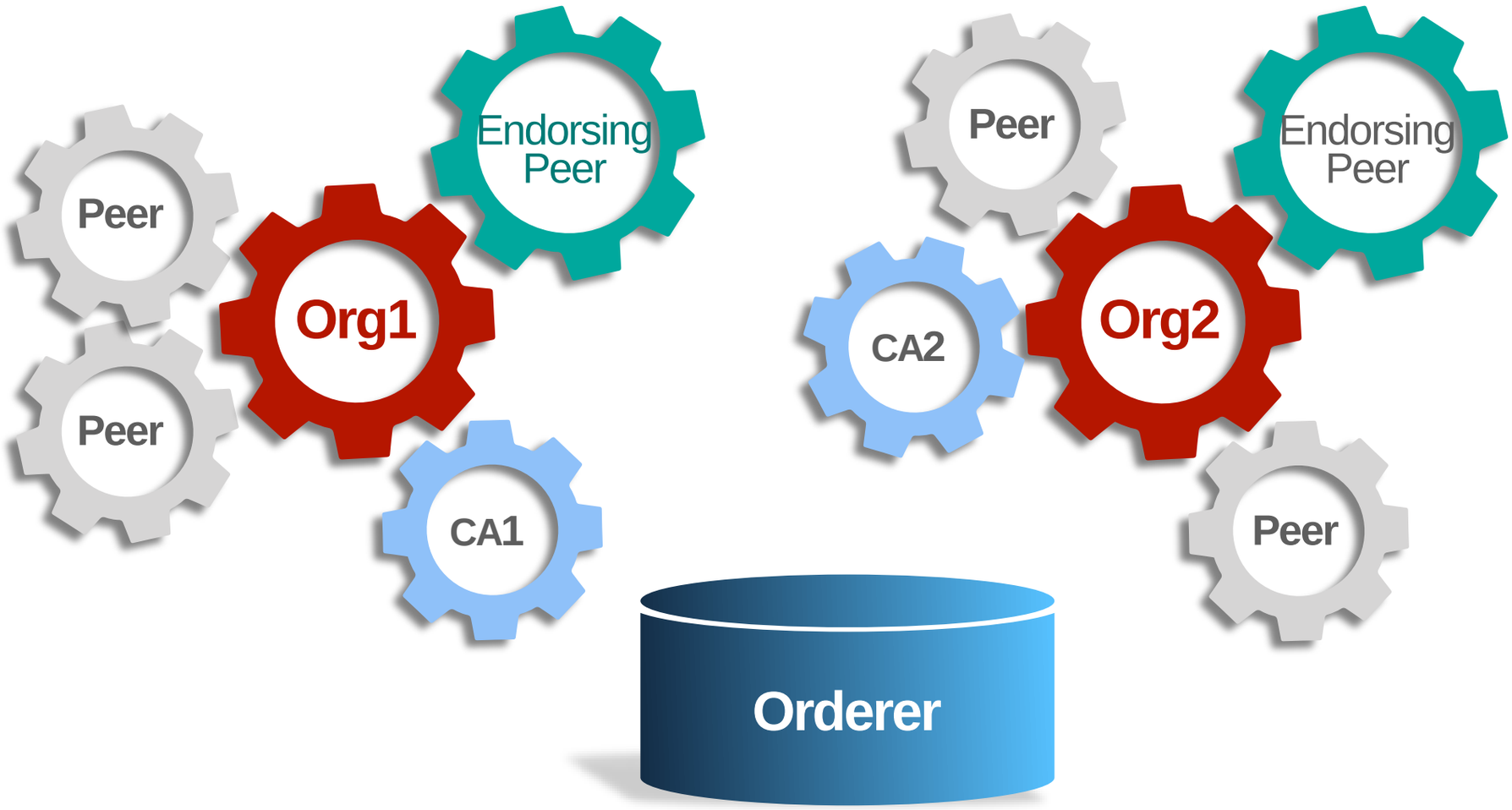
d **State database in CouchDB or LevelDB**

Hyperledger Fabric



Structure

- Ordering service
- Consensus algorithm
- Different organizations
- Scalability
- Certificate authorities
- Endorsing peers
- Execution of chaincode



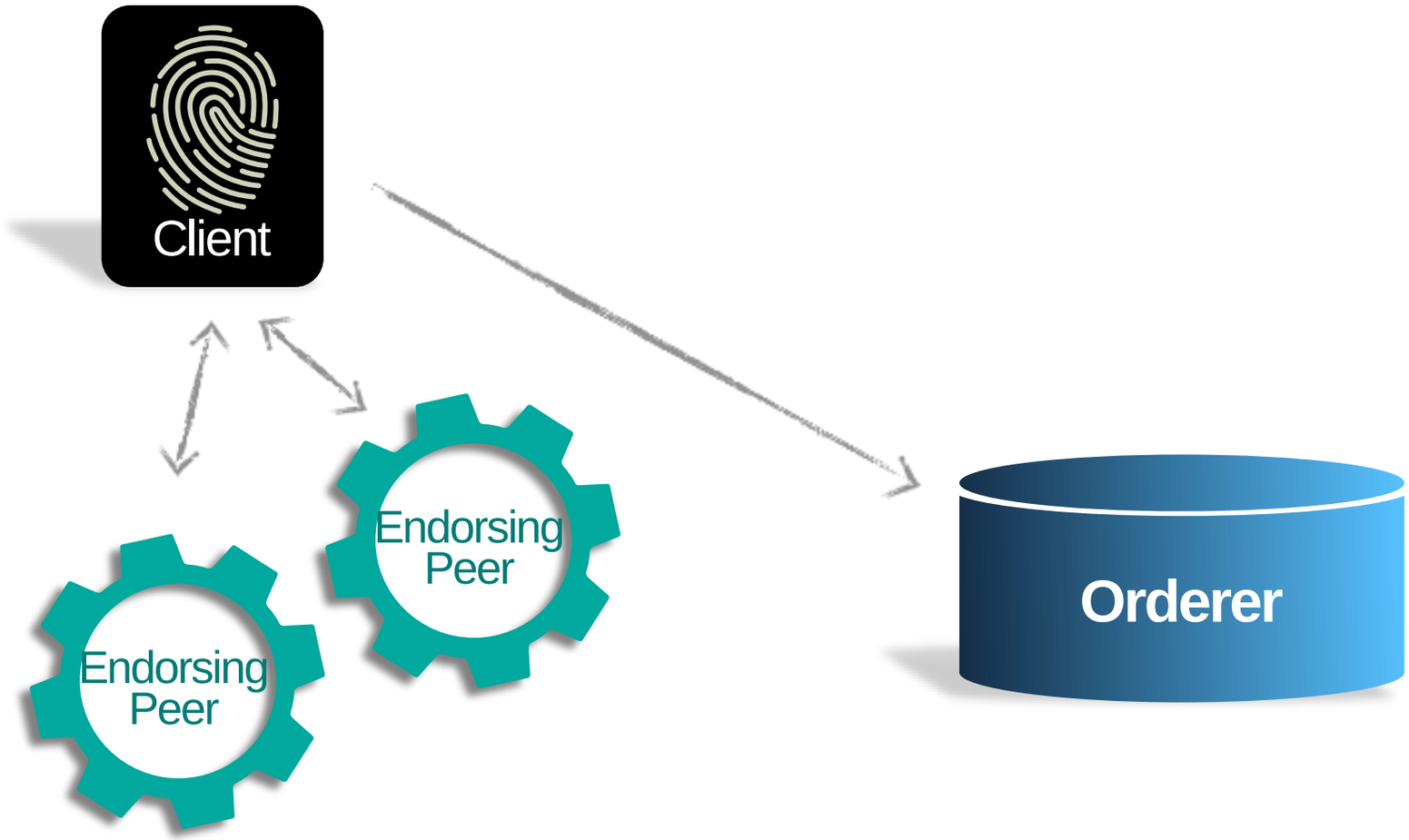
Execute

1. Client sends a transaction request
2. Execution of Chaincode
3. Sending the results back

Transaction Flow

Order

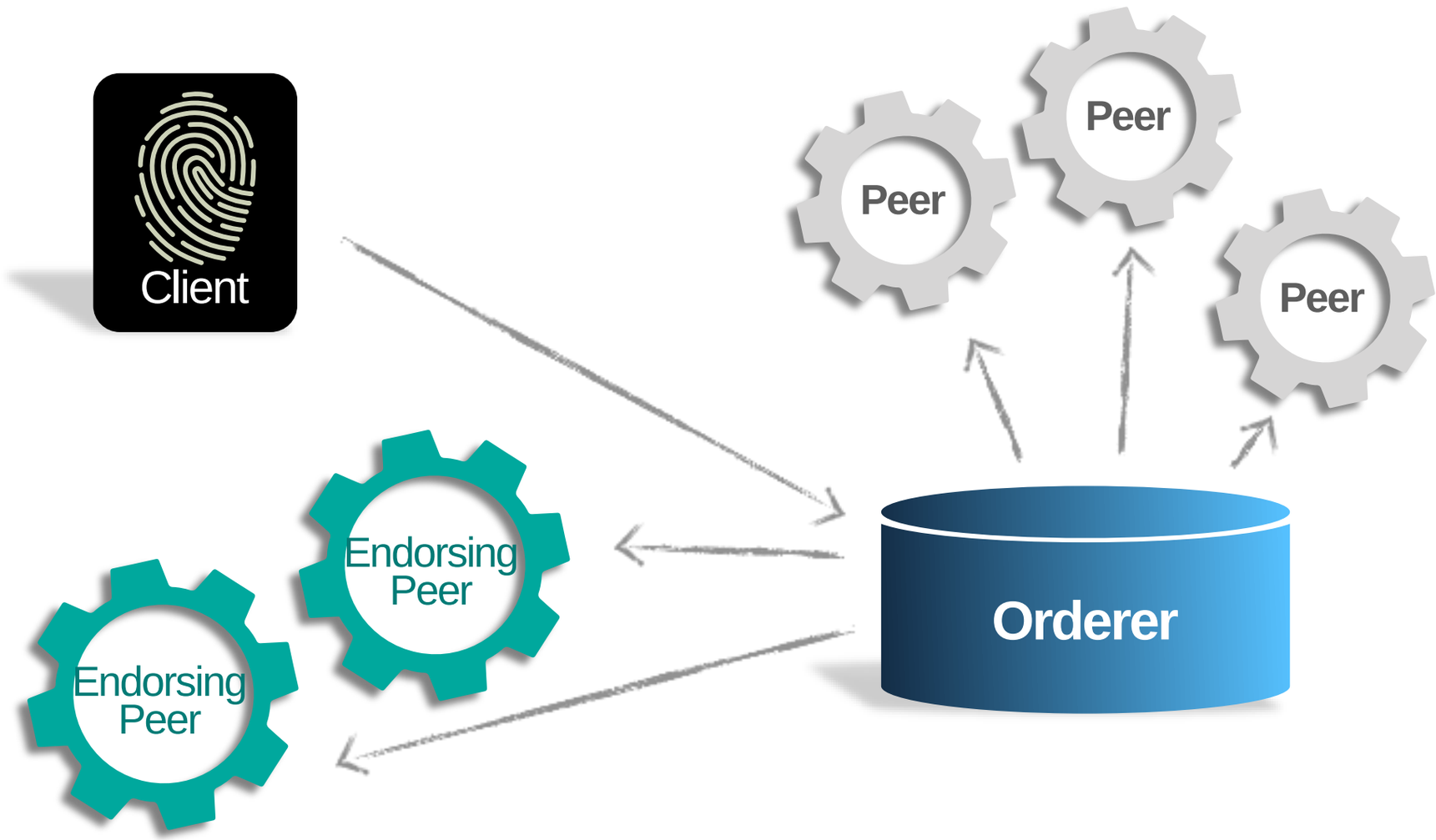
1. Results sent to the orderer
2. Ordering of transactions



Transaction Flow

Validate

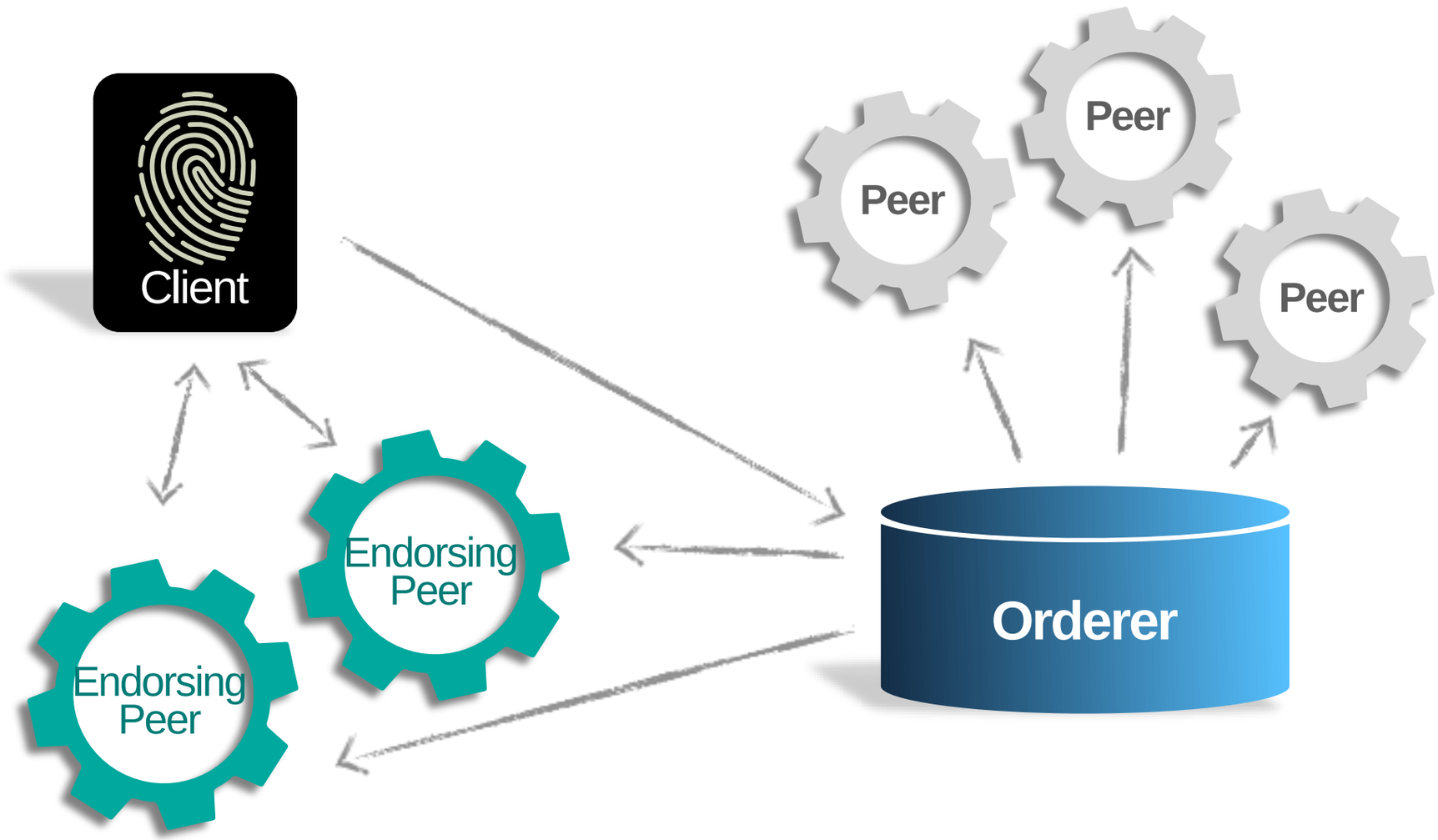
1. Validation of each block
2. If positive, it goes in the ledger



Transaction Flow

Validate

1. Validation of each block
2. If positive, it goes in the ledger



Transaction Flow

Framework over Fabric

1 Modeling language

- Participant
- Asset
- Transaction
- Event

2 Logic

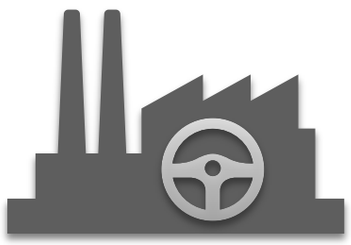
- Definition of the transaction logic
- Javascript code

3 Access Control

- Permissions on resources
- Complex conditioned rules

Hyperledger Composer

Model.cto



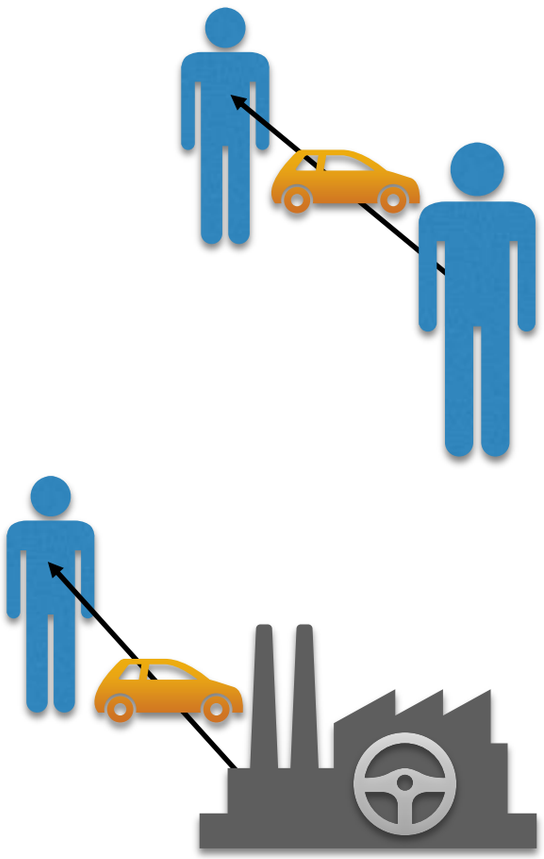
```
/**
* Definition of participants
*/
participant Person identified by username {
  o String username
  o String fullName
  o String email optional
  o String number optional
}

participant Manufacturer identified by makerId {
  o String makerId
  o String name
}
```

Model.cto

```
/**
* Definition of assets
*/
asset Vehicle identified by vin {
  o String vin
  o VehicleDetails vehicleDetails
  --> Person owner
}
```

Model.cto



```
/**
* Definition of transactions
*/
transaction ChangeOwner {
  --> Vehicle vehicle
  --> Person newOwner
}

transaction SellVehicle {
  o VehicleDetails vehicleDetails
  o String vin
  --> Person owner
}
```

Logic.js

```
/**
* Sell vehicle transaction
* @param {org.acme.vehicle.SellVehicle} arg
* @transaction
*/
function onSellVehicle (arg) {

}
```

Logic.js

```
const namespace = 'org.acme.vehicle';

// Extracting argument values

let vehicleDetails = arg.vehicleDetails;
const vin = arg.vin;
const owner = arg.owner;
const manufacturer = getCurrentParticipant();
vehicleDetails.manufacturer = manufacturer;
```

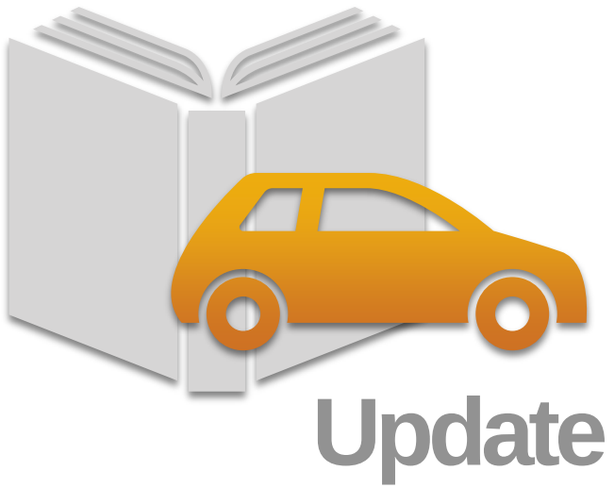
Logic.js

```
const factory = getFactory();

// Creating the vehicle
let vehicle =
    factory.newResource(namespace, 'Vehicle', vin)

vehicle.vehicleDetails = vehicleDetails;
vehicle.owner = owner;
```

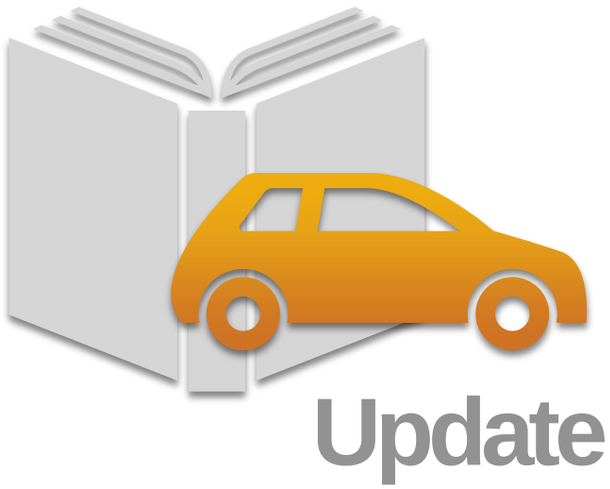
Logic.js




```
// Updating the registry

return getAssetRegistry(namespace + '.Vehicle')
    .then ( assetRegistry => {
        return assetRegistry.add(vehicle);
    })
```

Logic.js




```
// Updating the registry

return getAssetRegistry(namespace + '.Vehicle')
    .then ( assetRegistry => {
        return assetRegistry.add(vehicle);
    })
```

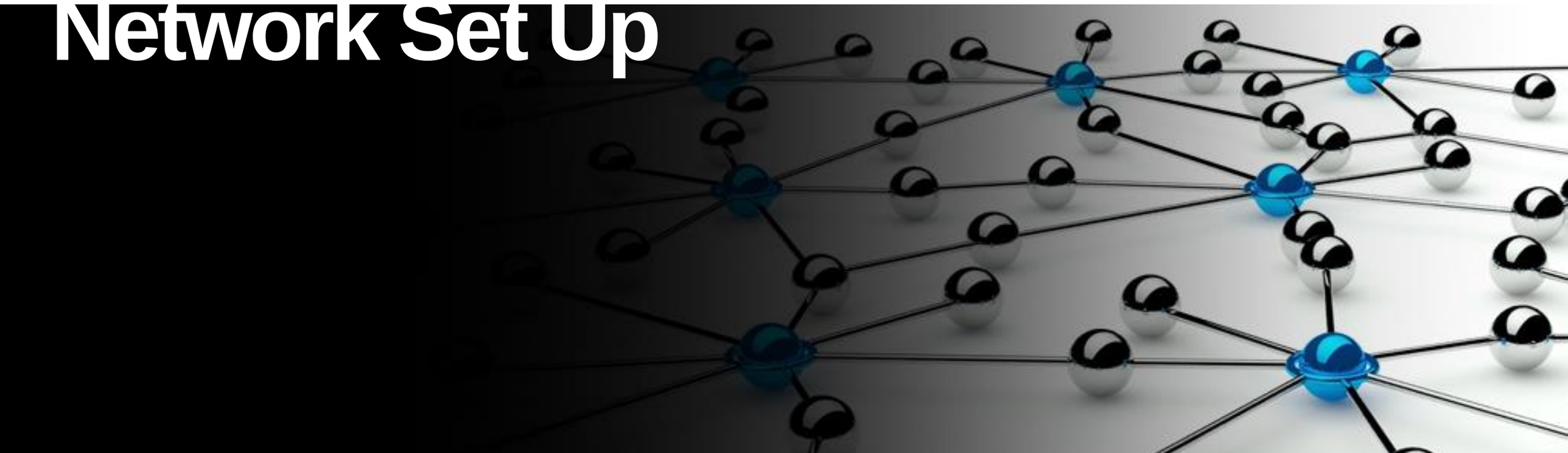
Permissions.acl

```
rule ManufReadsVehicle {
description: "Manuf may read a car only if he made it"
participant(manuf): "org.acme.vehicle.Manufacturer"
operation: READ
resource(vehicle): "org.acme.vehicle.Vehicle"
condition: ( manuf.getIdentifier() ==
    vehicle.vehicleDetails.manufacturer.getIdentifier() )

action: ALLOW
}
```

ORG1	ORG2	ORDERER
2 Peers	1 Peer	Solo
CA	CA	
CouchDB	CouchDB	

Network Set Up



Node 1	Node 2	Node 3	Node 4
Orderer	Org 1 Endorsing Peer	Org 1 Peer	Org 2 Endorsing Peer
	CouchDB	CouchDB	CouchDB
	Org1 CA		Org 2 CA
	SQL DB		SQL DB
	NodeJs App		NodeJs App

- APP exposed by NodeJs
- Composer SDK in NodeJs

Client

- Each element is a Docker container
- Docker Swarm to communicate

Opening connection

```
const BusinessNetworkConnection =
    require('composer-client').BusinessNetworkConnection;

module.exports = class Network {

    constructor(cardName) {
        this.cardName = cardName;
        this.connection = new BusinessNetworkConnection();
        this.definition;
        this.namespace = 'org.acme.vehicle';
    }

    async connect() {
        return this.definition =
            await this.connection.connect(this.cardName);
}
```

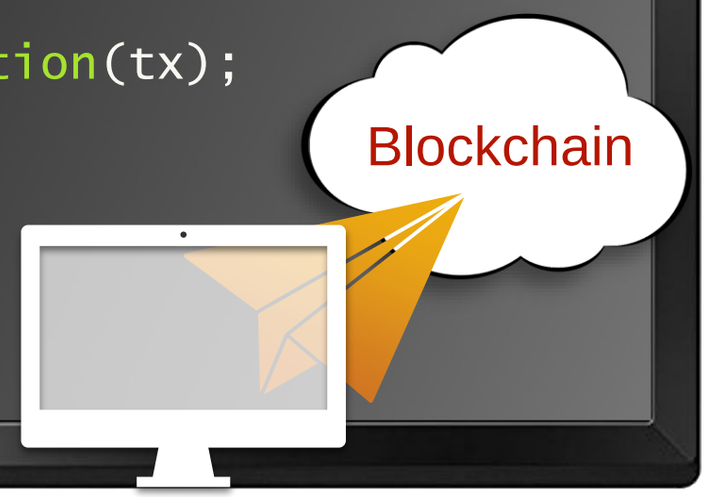
Submit Transaction

```
const factory = this.definition.getFactory();

// Submit transaction
const tx =
    factory.newTransaction(this.namespace, 'SellVehicle');

tx.vehicleDetails = vehicleDetalils;
tx.vin = vin;
tx.owner =
    factory.newRelationship(this.namespace, 'Person', ownerId);

return await this.connection.submitTransaction(tx);
```



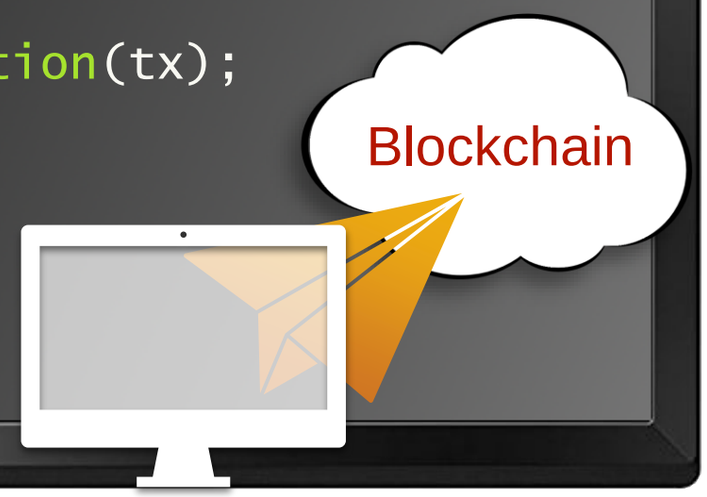
Submit Transaction

```
const factory = this.definition.getFactory();

// Submit transaction
const tx =
    factory.newTransaction(this.namespace, 'SellVehicle');

tx.vehicleDetails = vehicleDetalils;
tx.vin = vin;
tx.owner =
    factory.newRelationship(this.namespace, 'Person', ownerId);

return await this.connection.submitTransaction(tx);
```

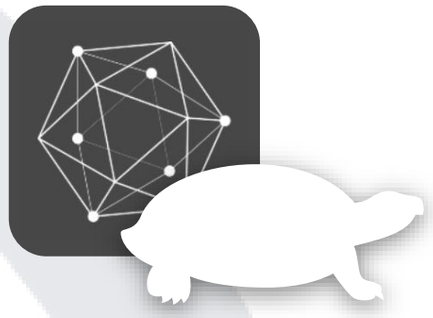


Repositories

Benchmark done with 1000 assets

GET —> ~ 10 seconds

PUT, DELETE —> ~ 100 SECONDS



- Improvement through Cloud deployment
- Not real-time applications

Performance evaluation

- Permissions in Hyperledger Composer
- Rich conditioned language

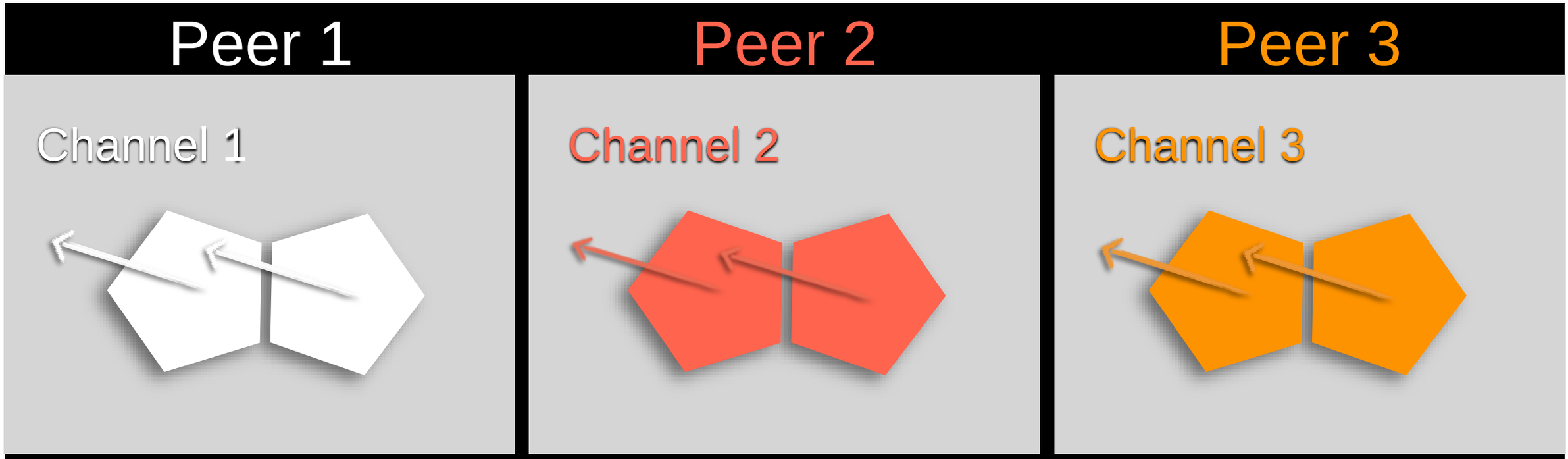
BUT…  The ledger remains in plain-text!

/var/hyperledger/production/ledgersData/chains/chains/{channel_name}/

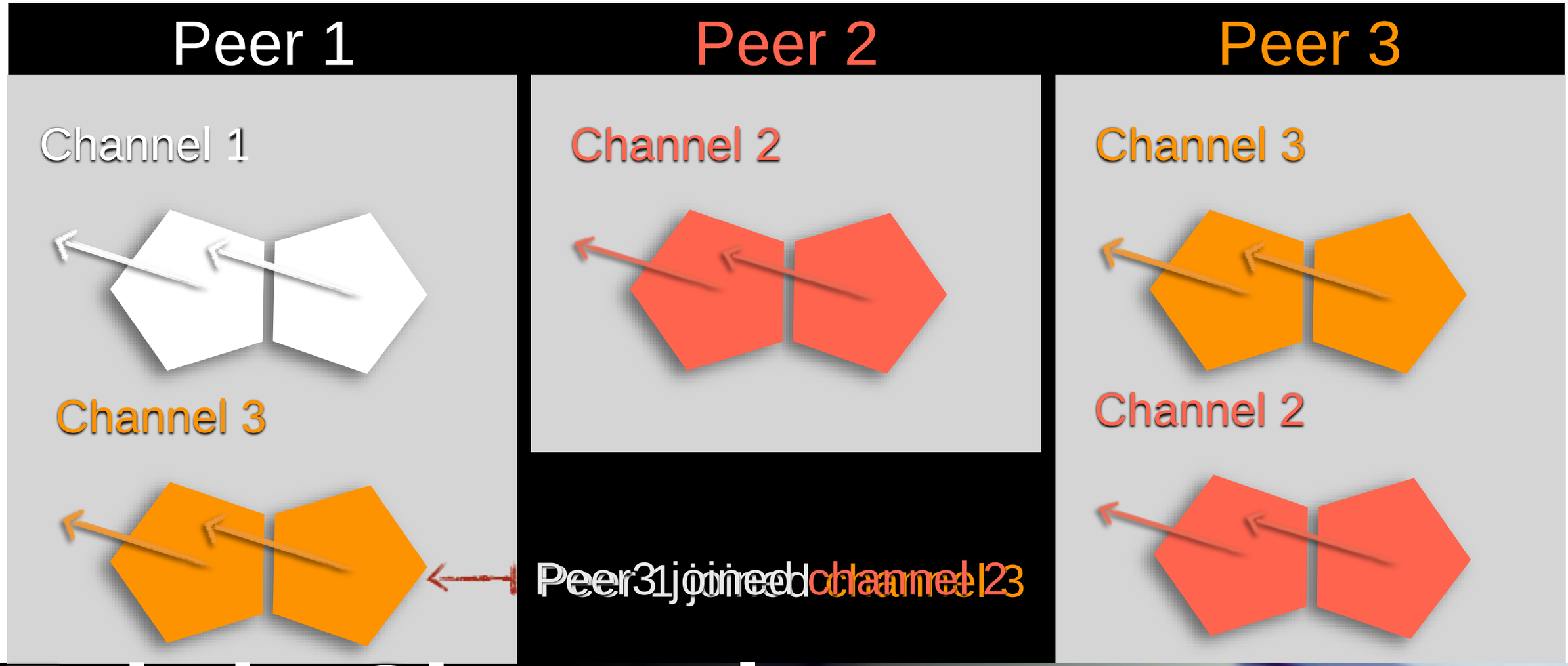
```
-Participant:org.acme.vehicle.Manufacturer1?{"$class":"org.acme.vehicle.Manufacturer","maker
Id":"1","name":"Manuf1","$registryType":"Participant","$registryId":"org.acme.vehicle.Manufa
cturer"}?
}Transaction:org.hyperledger.composer.system.AddParticipant526706ac841b2121309341e4ba105e764
```

PRIVACY

DO NOT DESPAIR!



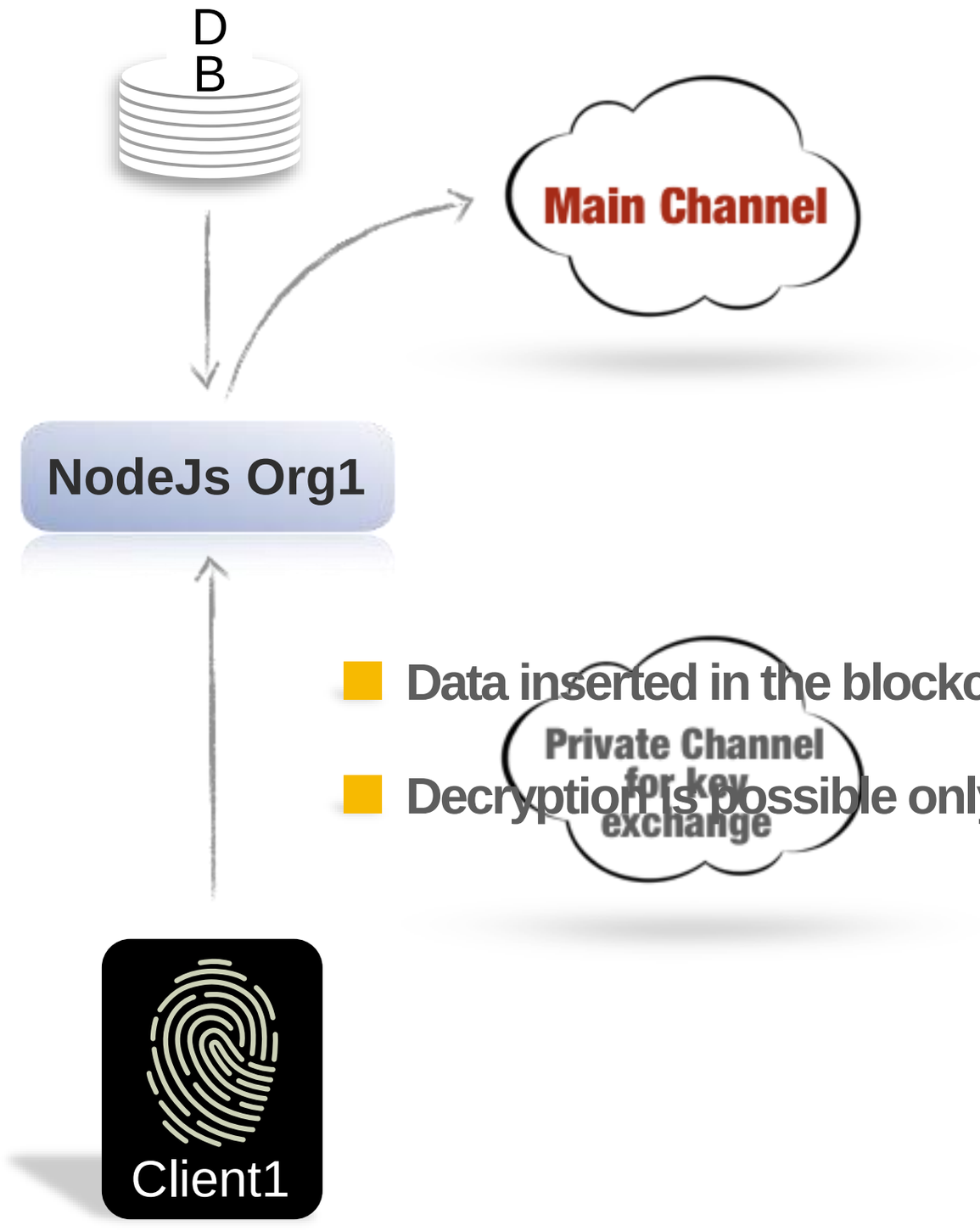
Fabric Channels



Fabric Channels

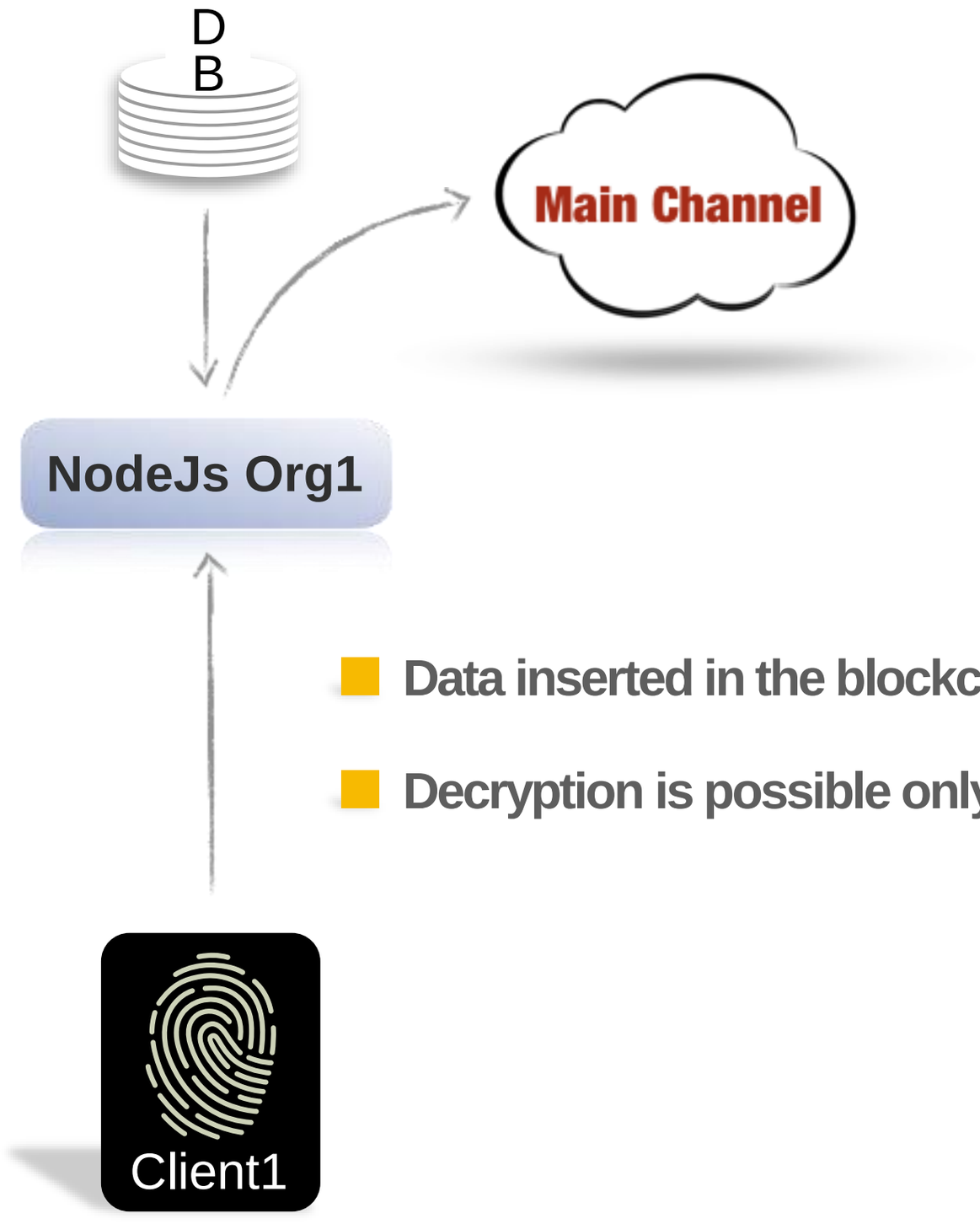
Off-chain encryption decryption

Off-chain encryption decryption



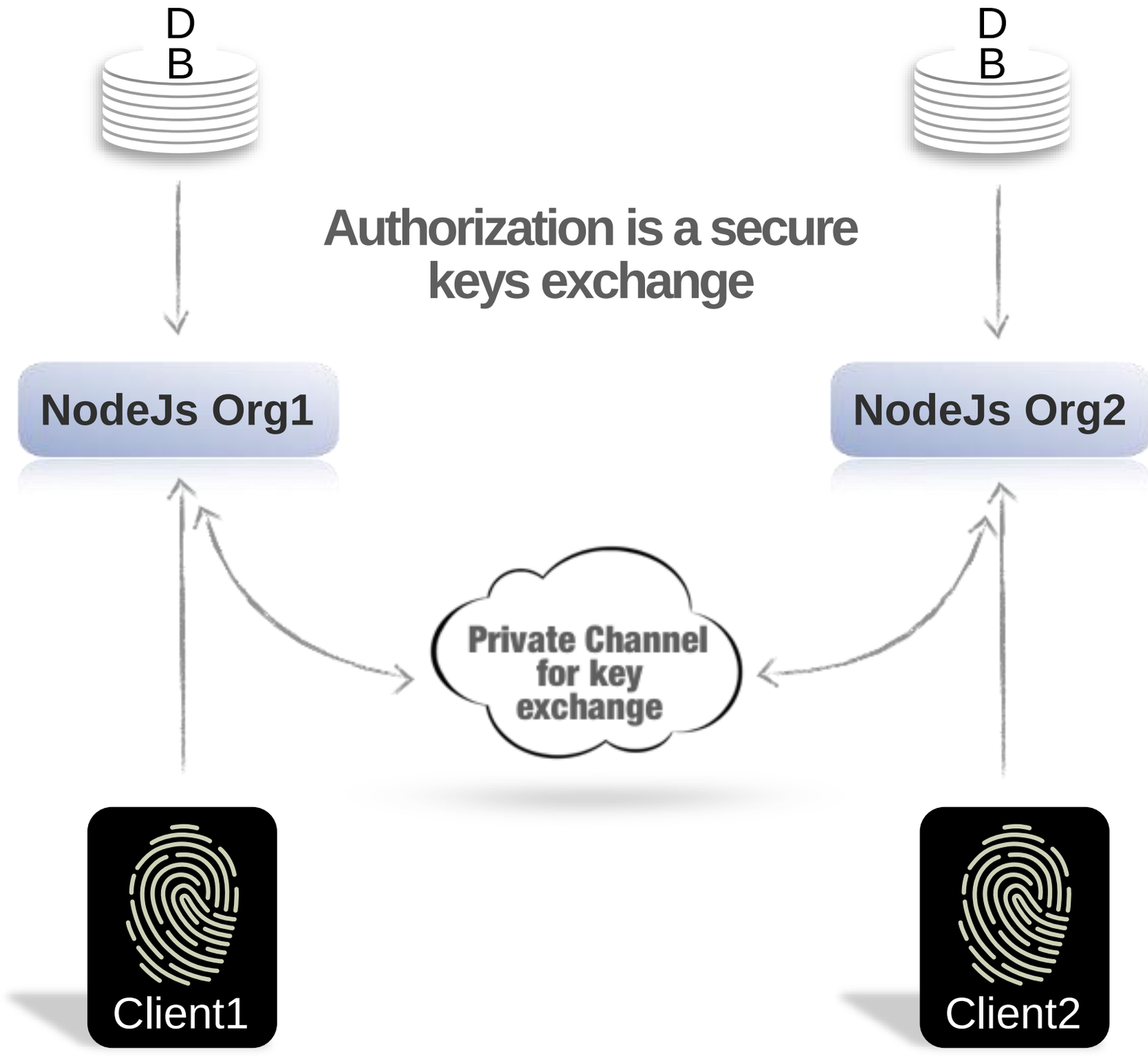
- Data inserted in the blockchain is encrypted
- Decryption is possible only with the keys

Off-chain encryption decryption

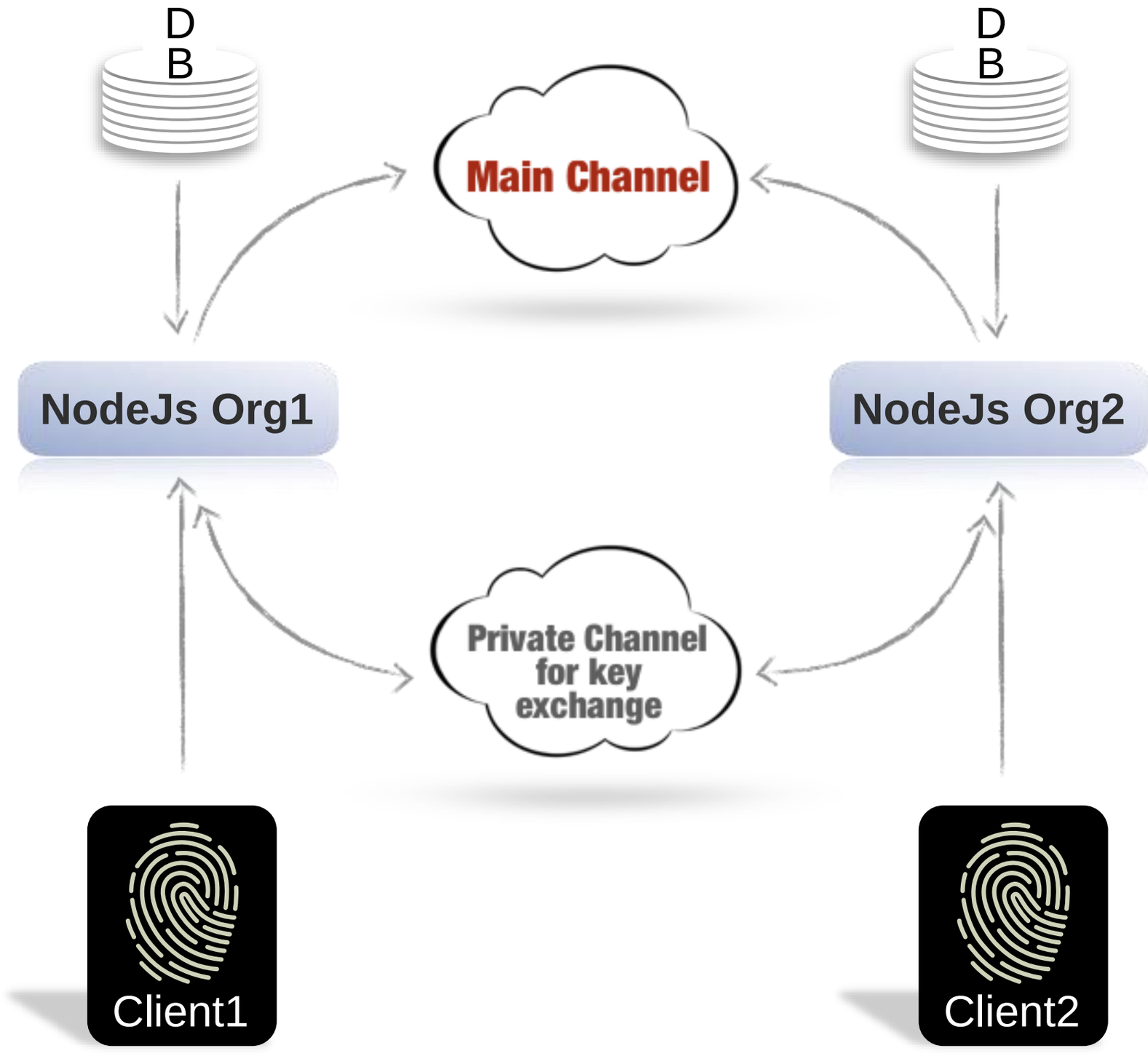


- Data inserted in the blockchain is encrypted
- Decryption is possible only with the keys

Off-chain encryption decryption



Off-chain encryption decryption



a Companies may share their data securely

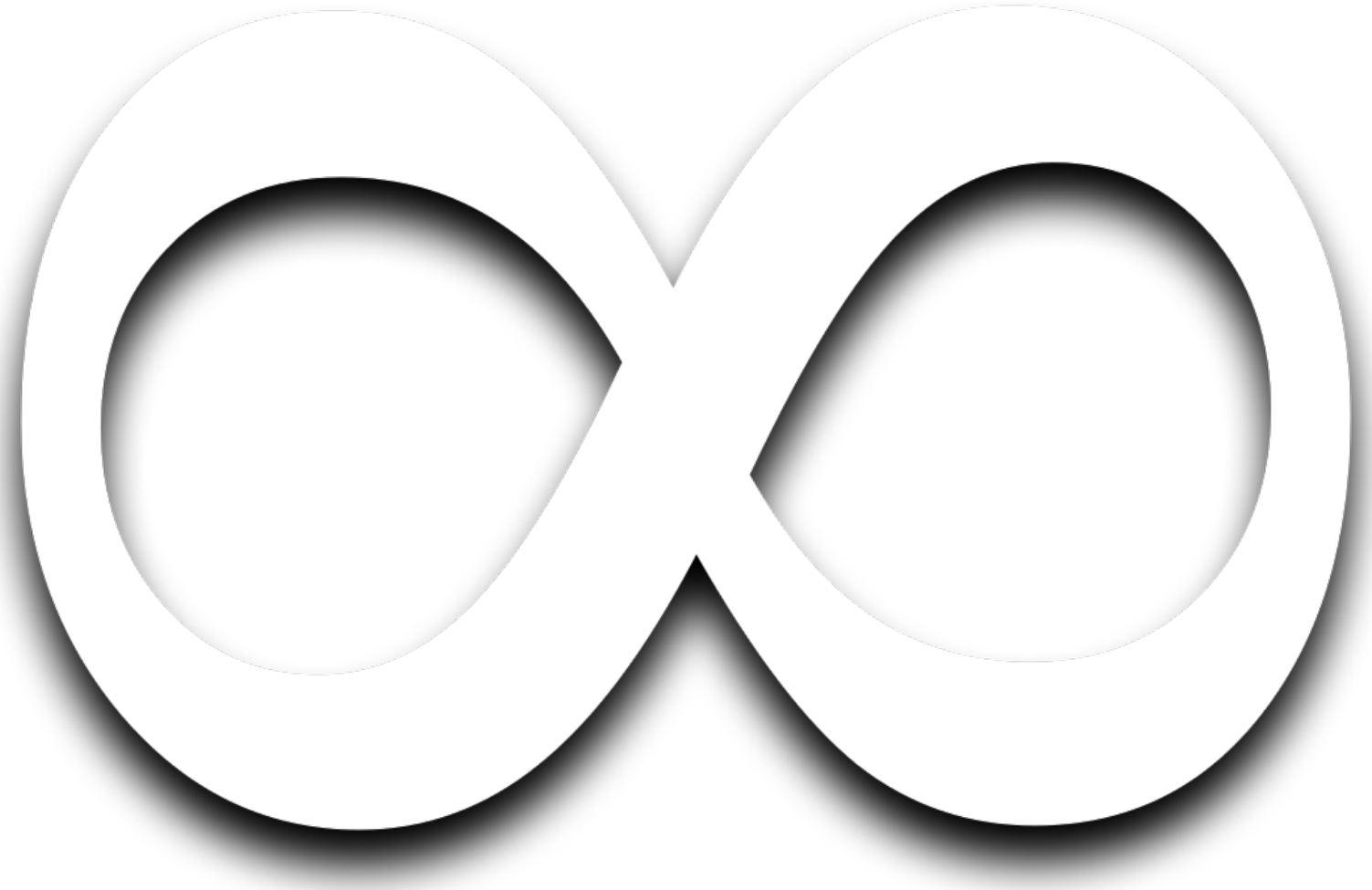
b They can rely on a robust architecture

c Each company checks directly wether the structure of the network is respected

d Performances may be improved through Cloud deployment or container optimizations

e Privacy must be taken into account in order to separate data

CONCLUSIONS



Endless Possibilities

edoardoconte