Hyperledger Fabric

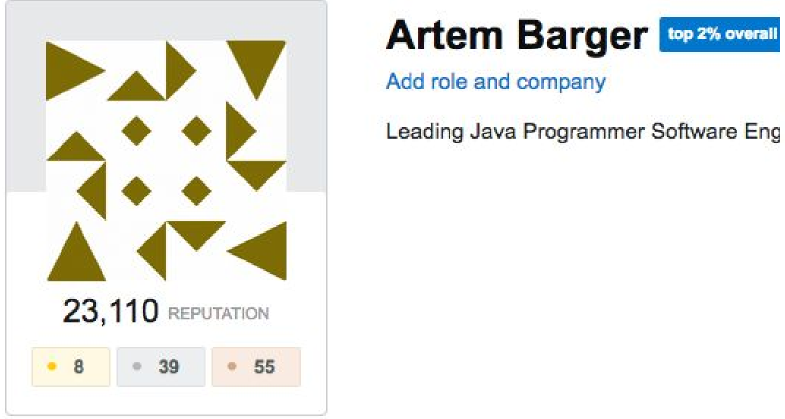
the architecture of the permissioned ledger

Artem Barger

(bartem@il.ibm.com)

About me…

- IBM Haifa Lab Cloud Foundation Research
- 10+ years of experience in design and development of distributed system
- Maintainer of Linux Foundation Hyperledger Project
- Decent background in Java server side devlopment
- ASF Committer (Apache Commons)



Artem Barger top 2% overall

Add role and company

Leading Java Programmer Software Eng

23,110 REPUTATION

● 8 ● 39 ● 55

C0rWin

@C0rWin

Tweets	Following	Followers
790	341	70

Outline

- Blockchain
  - ✓ Basic concepts
  - ✓ Permissionless (Bitcoint, Etherium) vs Permissioned (Hyperledger)

- Hyperledger
  - ✓ Previous architecture
  - ✓ Driving towards more scalable and secure architecture
  - ✓ Better privacy
  - ✓ Identity Management

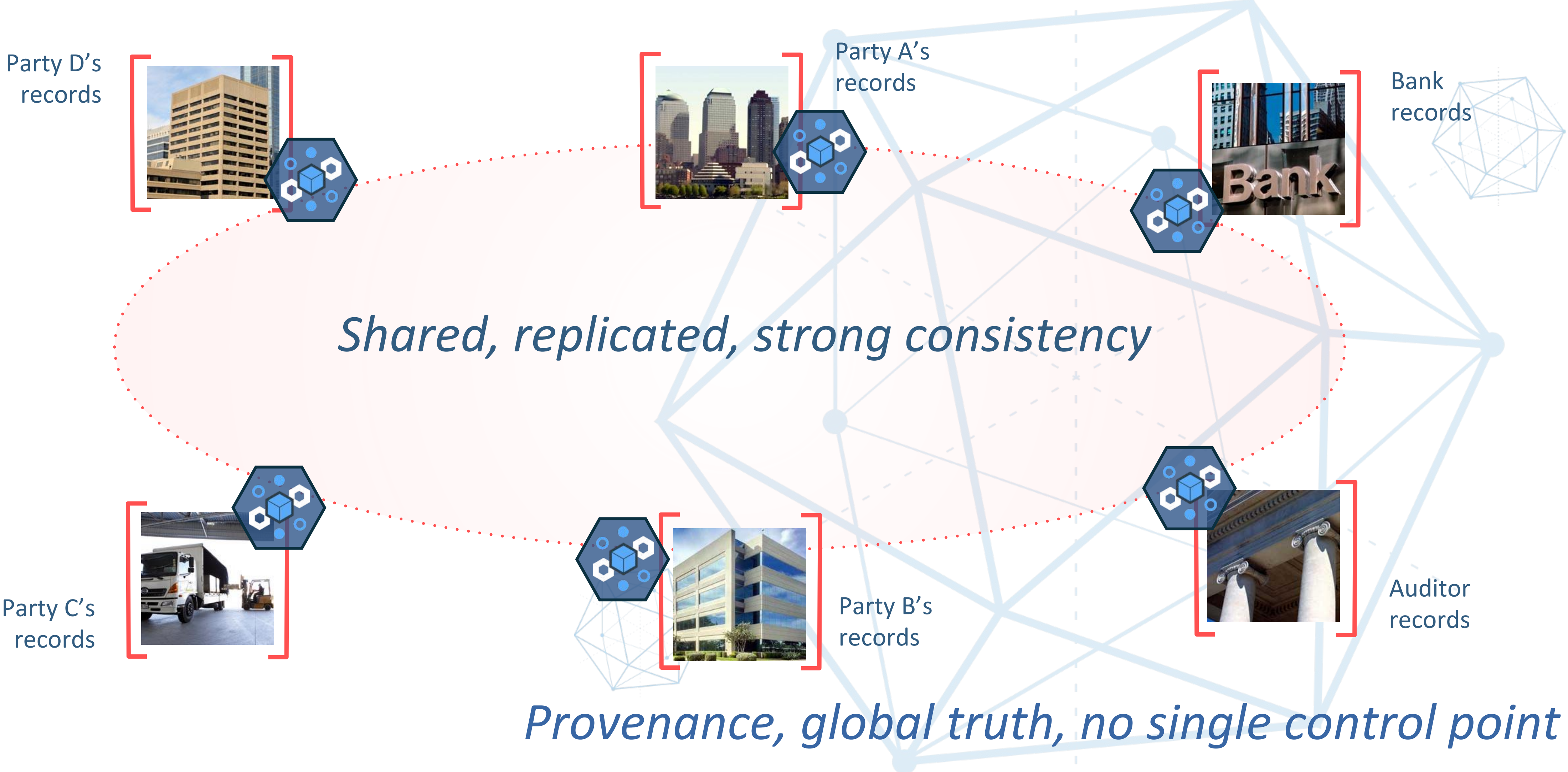
- QA

Blockchain

I THINK WE SHOULD BUILD A BLOCKCHAIN
UH-OH
E-mail: SCOTTADAMS@AOL.COM
S. Adams
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1995 United Feature Syndicate, Inc.(NYC)
11/17
WHAT COLOR DO YOU WANT THAT BLOCKCHAIN?
I THINK MAUVE HAS THE MOST RAM.

What is blockchain?

Blockchain: A shared distributed ledger allowing participants in a business network to work with one system of record

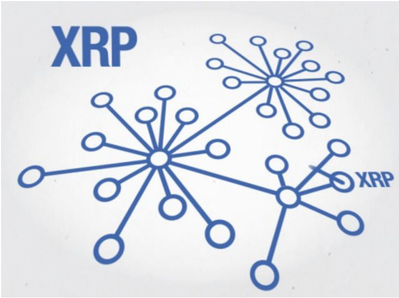


Provenance, global truth, no single control point

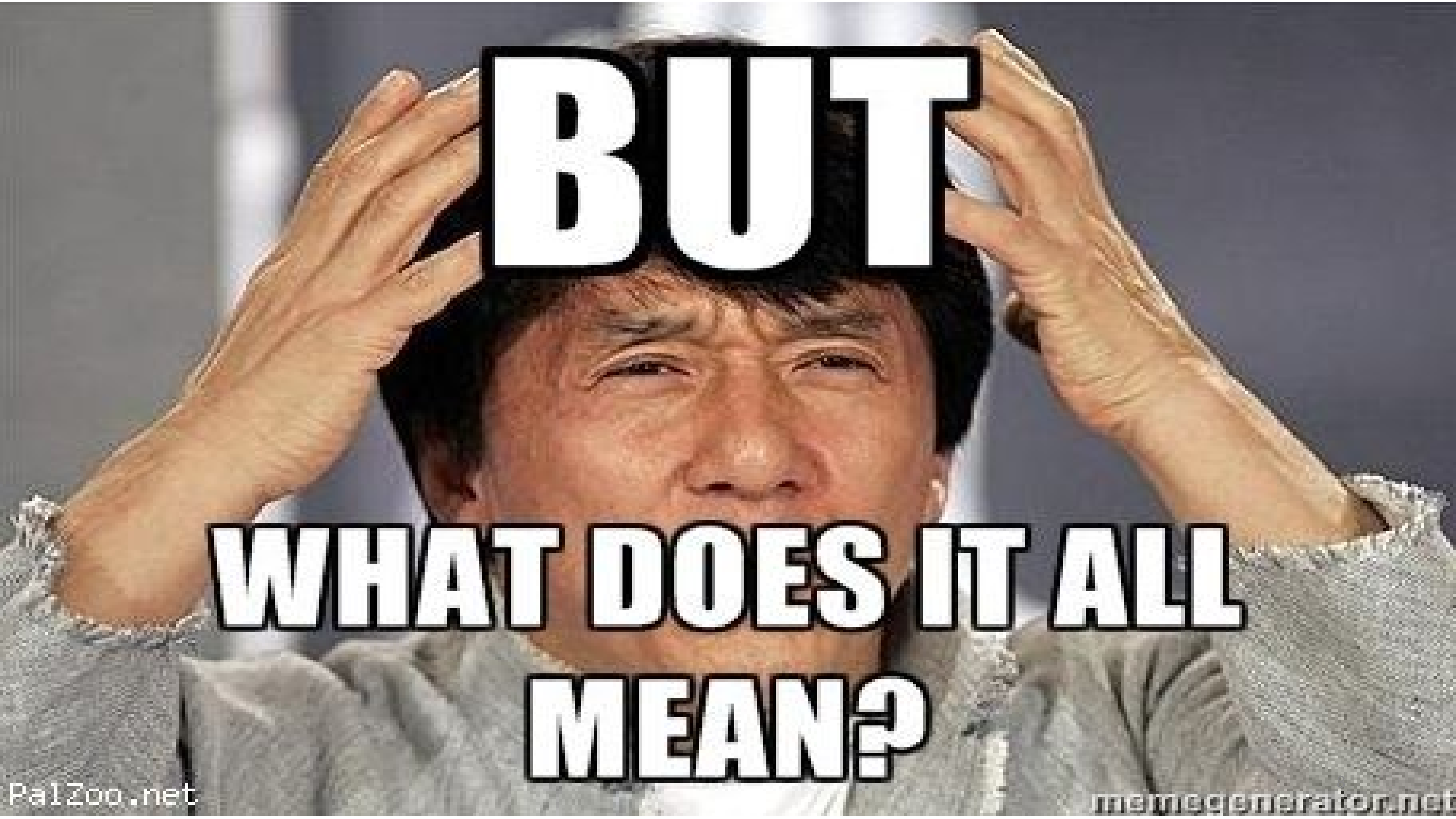
Blockchain

- Introduced in 2008 [Bitcoin08]
- **Decentralized** networks to decide on the order in which network **transactions** are **validated** & append to a system wide ledger
    - **Decentralized**: network controlled by independent entities
    - **Transactions**: messages announced across the network
    - **Validity**: following specified set of rules

ETHEREUM



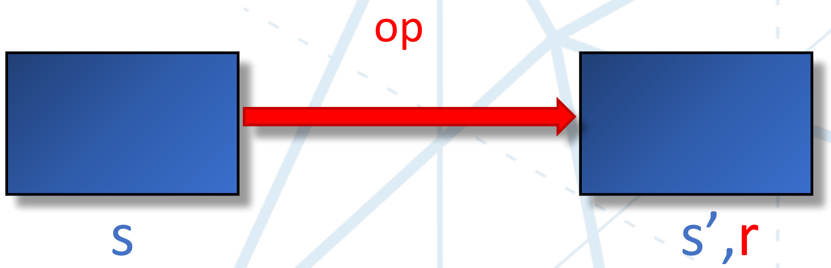
ripple


BUT
WHAT DOES IT ALL MEAN?

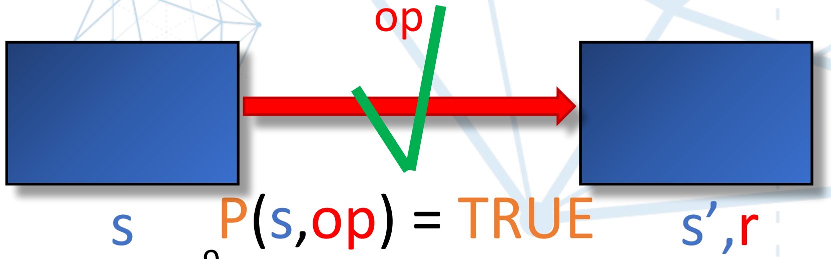
A state machine

- Define a functionality F
  - ✓ Operation op transforms a state s to new state s’ and generates response r

$F(s,op) \rightarrow (s',r)$

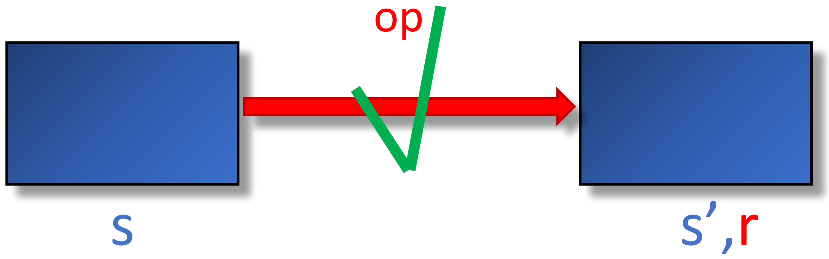


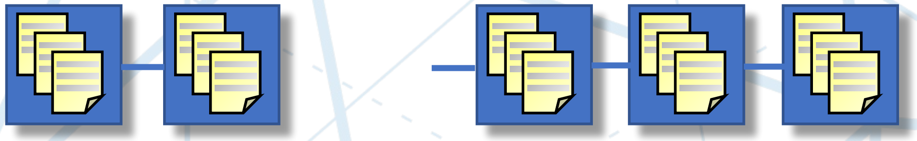
- Validation
  - ✓ Operation has to be valid according to a predicate P



Blockchain state machine

- Append-only log
  - ✓ Every operation op appends a ”block” of valid transactions to the log

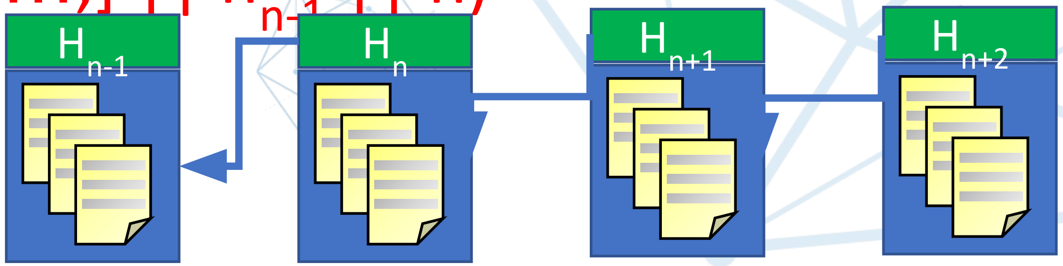




- Log content is verifiable from the most recent element

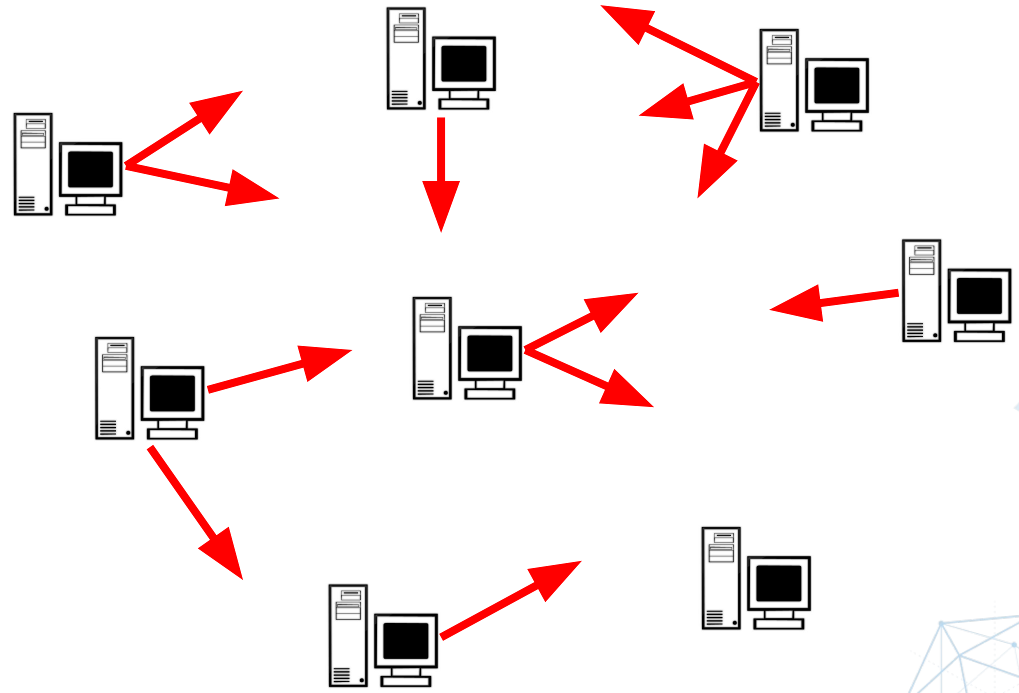
- Log entries form a hash chain:

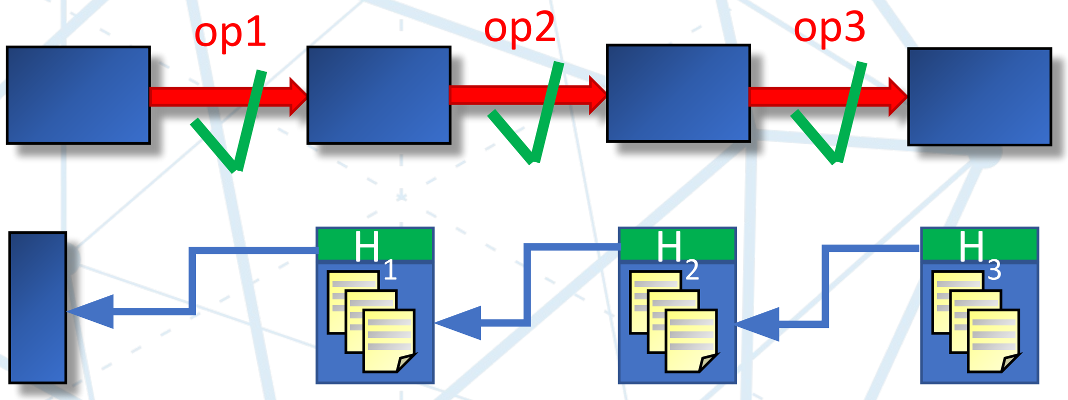
  $h_n \leftarrow \text{Hash}([tx_1, tx_2, \ldots,] \,||\, h_{n-1} \,||\, n)$



Distributed peer-to-peer protocol to create a ledger

Nodes produce new transaction





Nodes run a protocol to construct the ledger

Blockchain protocol features

- Only ”valid” operations (transactions) are “executed”

- Primitive transactions such as in Bitcoin
  - ✓ Statement of ownership for crypto coins:
    “X amount of bitcoins belongs to Y” signed by Z

- More complex transactions (AKA smart contracts == arbitrary code)
  - ✓ Encapsulate business logic that responds to events (on blockchain) and may produce response by for example transferring asset
  - ✓ Auction, elections, trading, investment decision, supply chains, etc…

Blockchain security

- Transactional privacy
  - ✓ Anonymity or pseudonymity through cryptographic tools
- Smart contracts privacy
  - ✓ Distributed secure computations on encrypted data
    - ZKP, Homomorphic encryption
- Accountability & non-repudaiation
- Auditability & transperancy
  - ✓ Hash chain

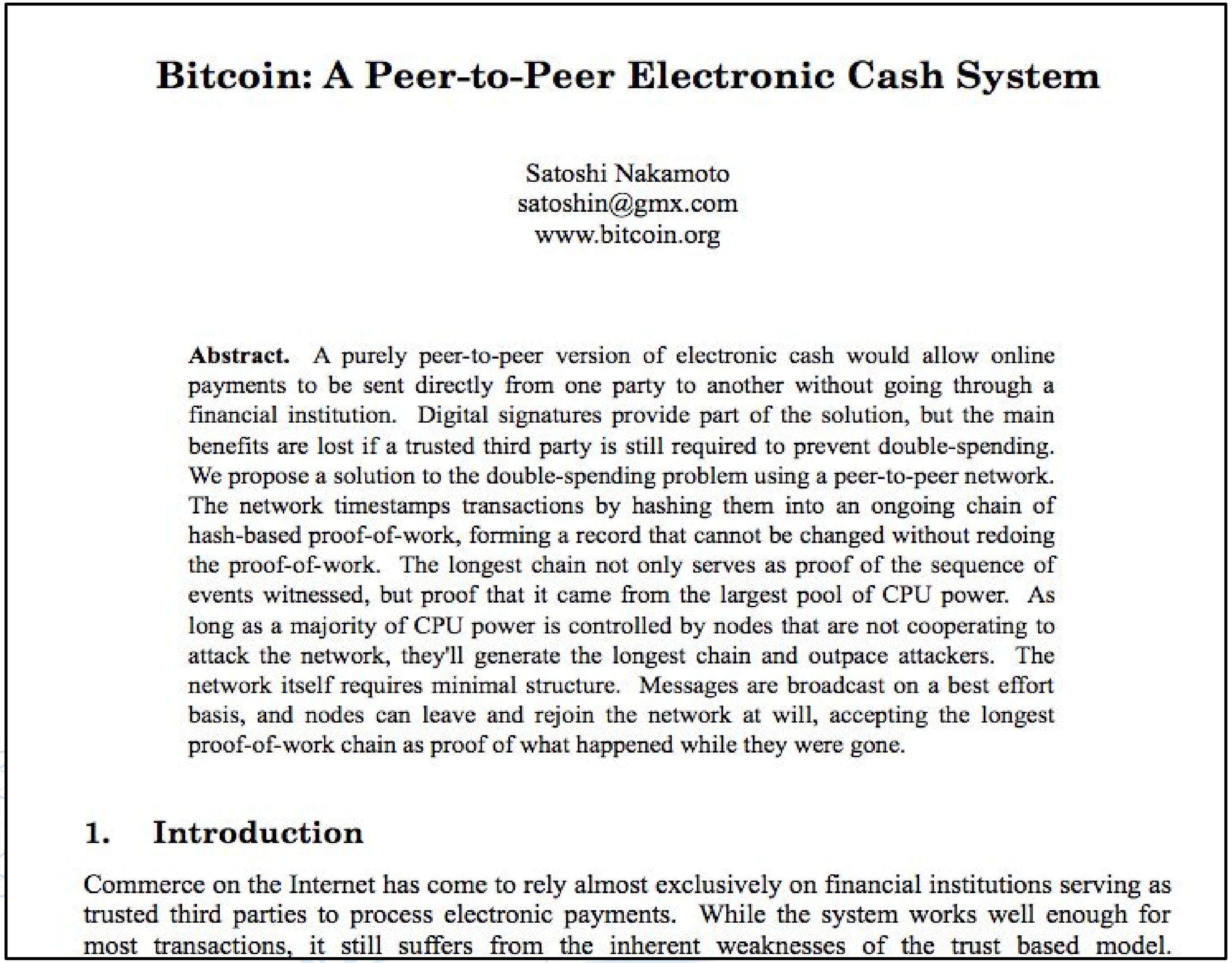
Conflict
Resolution

Nakamoto consensus - Bitcoin

- Nodes prepare blocks
  - ✓ List of transactions (tx)
  - ✓ All transactions valid

- Lottery race
  - ✓ Solve hard crypto puzzle
  - ✓ Select an arbitrary winner
  - ✓ Winner block applied to the ledger

- All nodes verify and validate new block
  - ✓ Longest hash chain wins



Bitcoin: A Peer-to-Peer Electronic Cash System

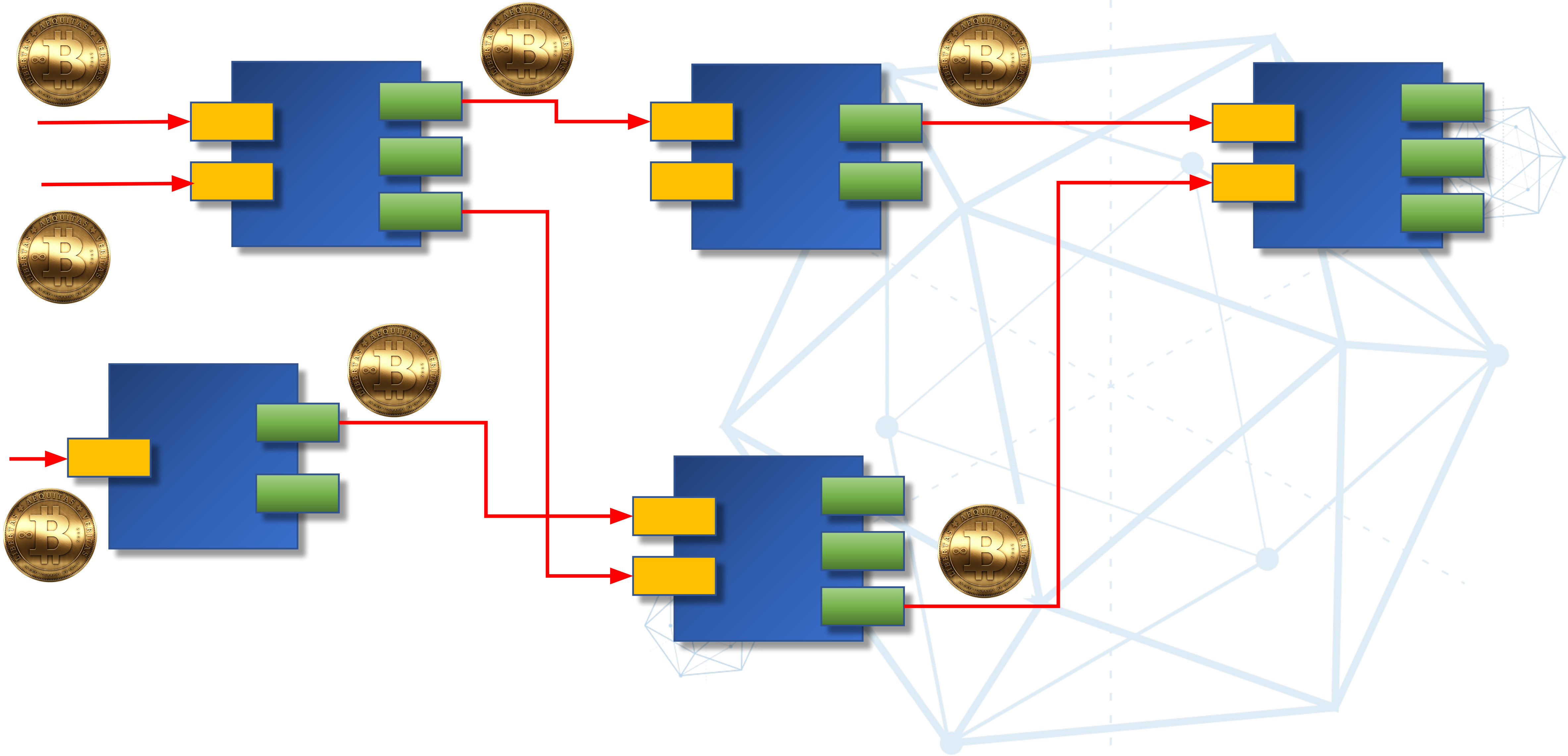
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

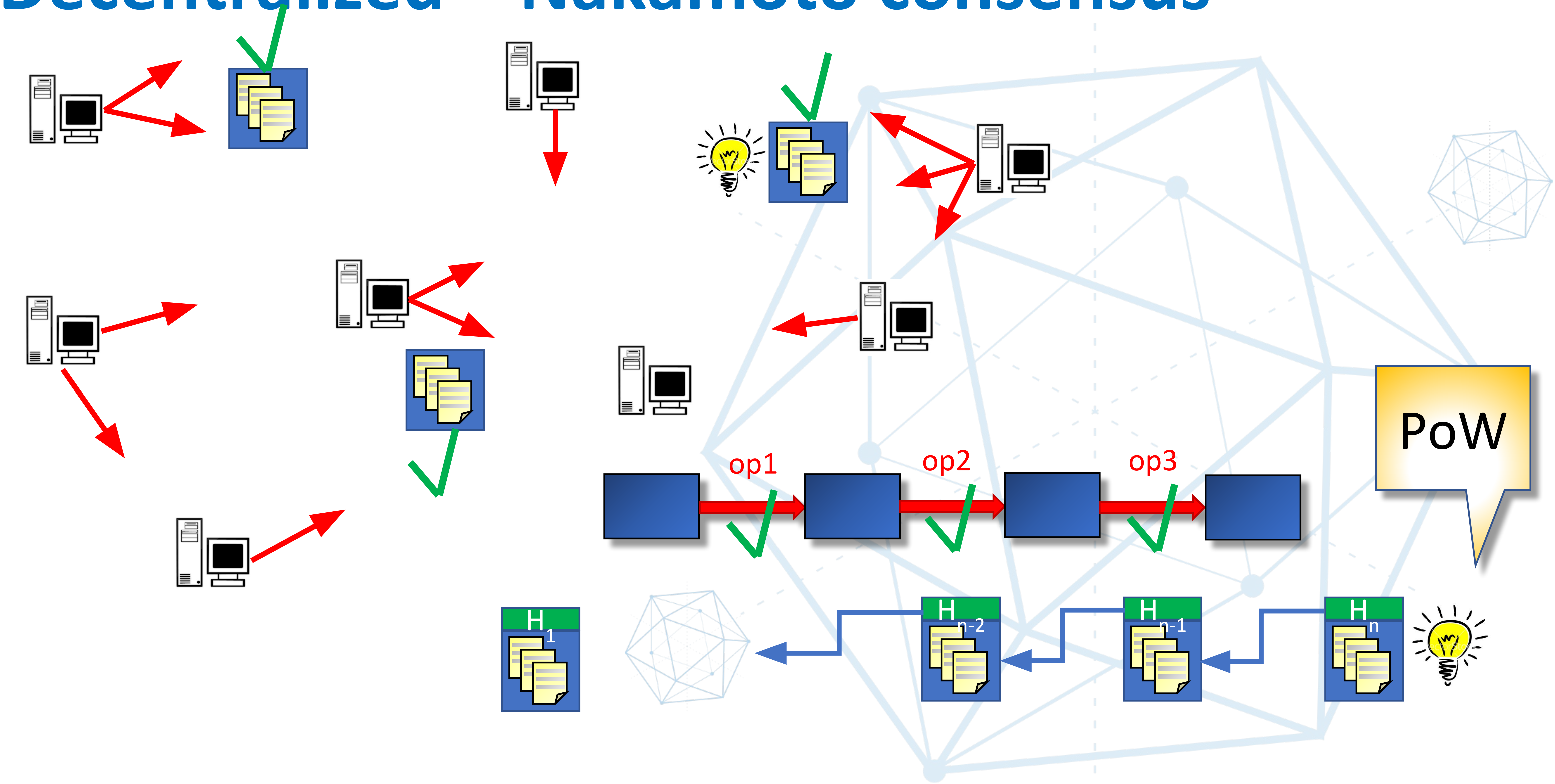
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model.

Bitcoin transactions (UTXO)



Decentralized – Nakamoto consensus



Decentralized = permissionless

- Resilient to censorship and suppression
  - ✓ No central entity

- Nakamoto consensus ⟶ proof-of-work (PoW)
  - ✓ Once CPU, one vote
  - ✓ Majority of hashing power could control the network
  - ✓ Mining, gives incentive to participate

- Protocol features
  - ✓ Stability is a tradeoff between dissemination of new block and mining rate
  - ✓ Decisions are not final (have to wait for 6 new blocks before confirming tx)

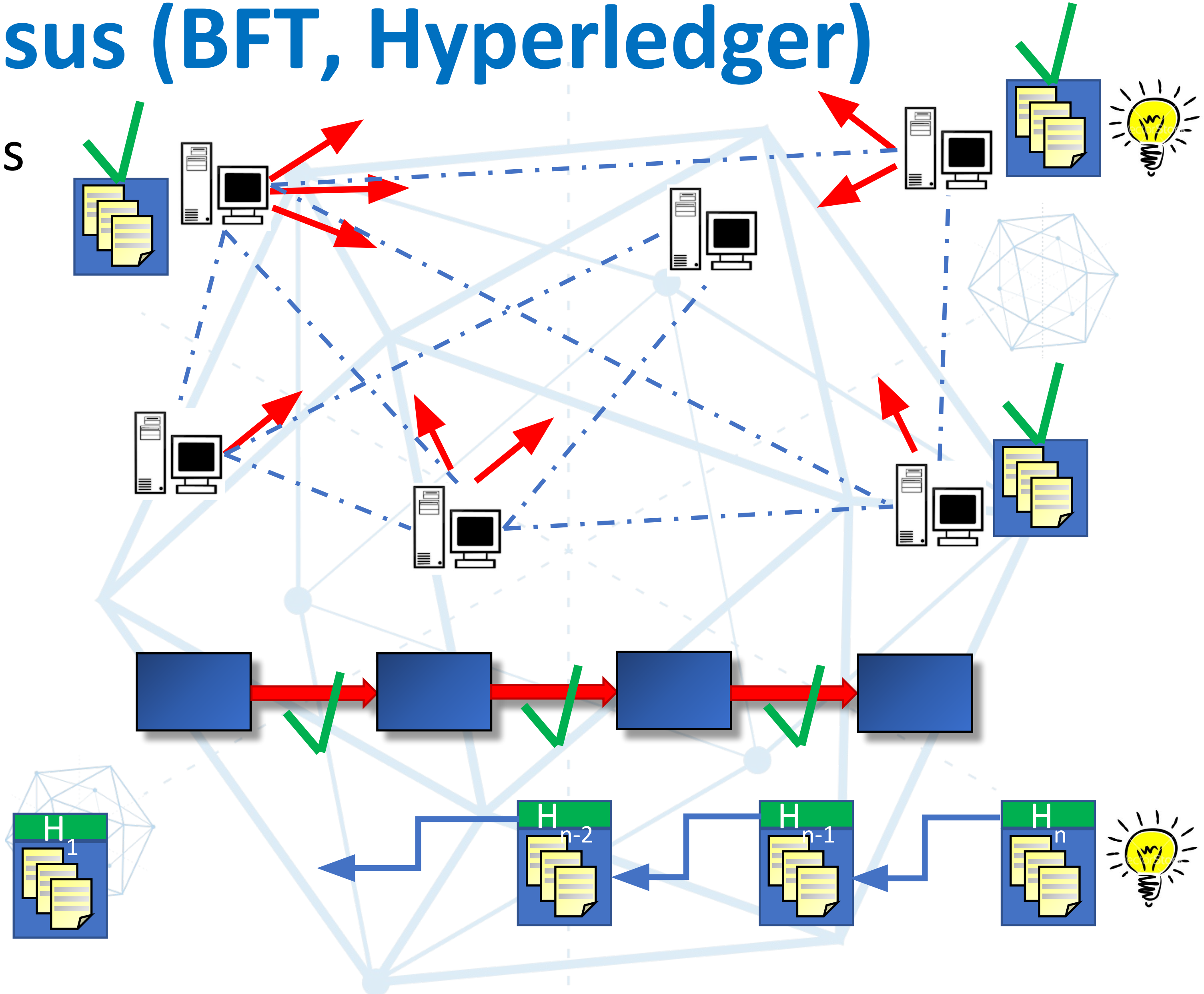
Consortium consensus (BFT, Hyperledger)

- Designed set of homogeneous validator nodes

- Byzantine agreement
  - Generalized quorums

- Tx sent to consensus nodes

- Consensus validates, decides and disseminates results



© IBM Corporation, 2017

Artem Barger (bartem@il.ibm.com)

Consortium consensus = permissioned

- Central entity controls group membership
  - ✓ Dynamic membership changes in protocol
  - ✓ Membership may be decided inline, by protocol itself

- Features
  - ✓ BFT and consensus are very-well understood problem
  - ✓ Many systems already provide crash tolerant consensus (Chubby, ZK, etcd)
  - ✓ Requires n^2 communication (might work for 1—100 nodes, fails for 1000)

- Revival of research in BFT protocols
  - ✓ Focus on scalability and communication efficiency

Permissioned vs Permissionless blockchains

Permissionless

- **Don’t trust anyone!**
    - Required PoW => Slow!
- Miners maintains the network
- If more than 51% controlled by one group the entire network might be hacked
- On-chain assets
- Censorship resistant

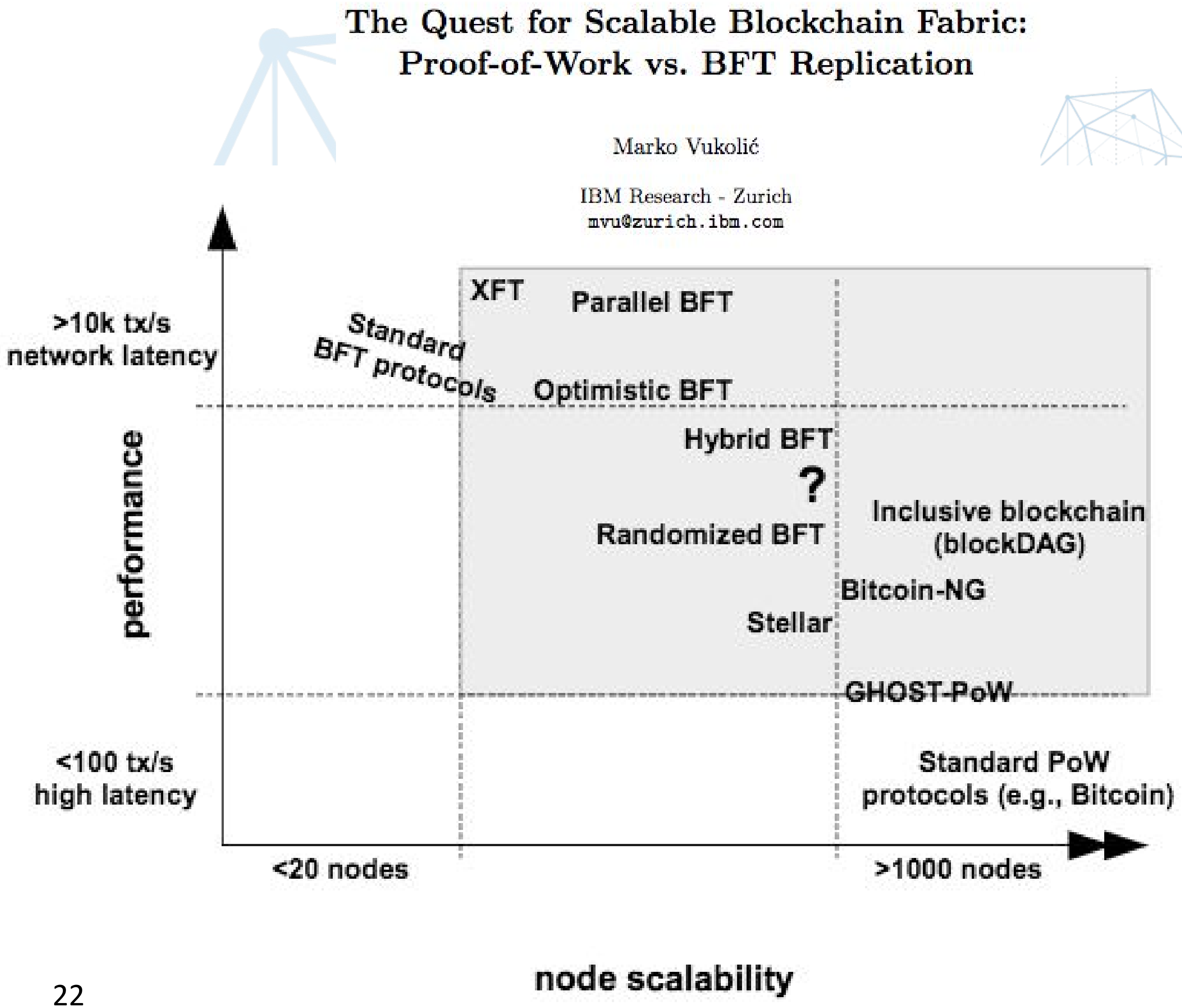
Permissioned

- **Trust everyone!**
    - Faster, only requires validation for agreement
- Need central administrator to control network entities
- Off-chain assets
- Better irreversibility and control

Scalability-performance tradeoff

- PoW offers good scalability with very poor performance

- BFT offers good performance for small number of replicas



The Quest for Scalable Blockchain Fabric:
Proof-of-Work vs. BFT Replication

Marko Vukolić

IBM Research - Zurich
mvu@zurich.ibm.com

What is the HYPERLEDGER PROJECT?

Open source collaborative effort to advance cross-industry blockchain technologies.

Hosted by The Linux Foundation

Global collaboration including leaders in finance, banking, IoT, supply chain, manufacturing and technology

Hyperledger Project Members

Premier

- accenture High performance. Delivered.
- AIRBUS
- CME Group
- DEUTSCHE BÖRSE GROUP
- Digital Asset
- DTCC Securing Today. Shaping Tomorrow.
- FUJITSU
- HITACHI Inspire the Next
- IBM
- intel
- J.P.Morgan
- R3
- 万达·飞凡科技 WANDA FFAN TECHNOLOGY

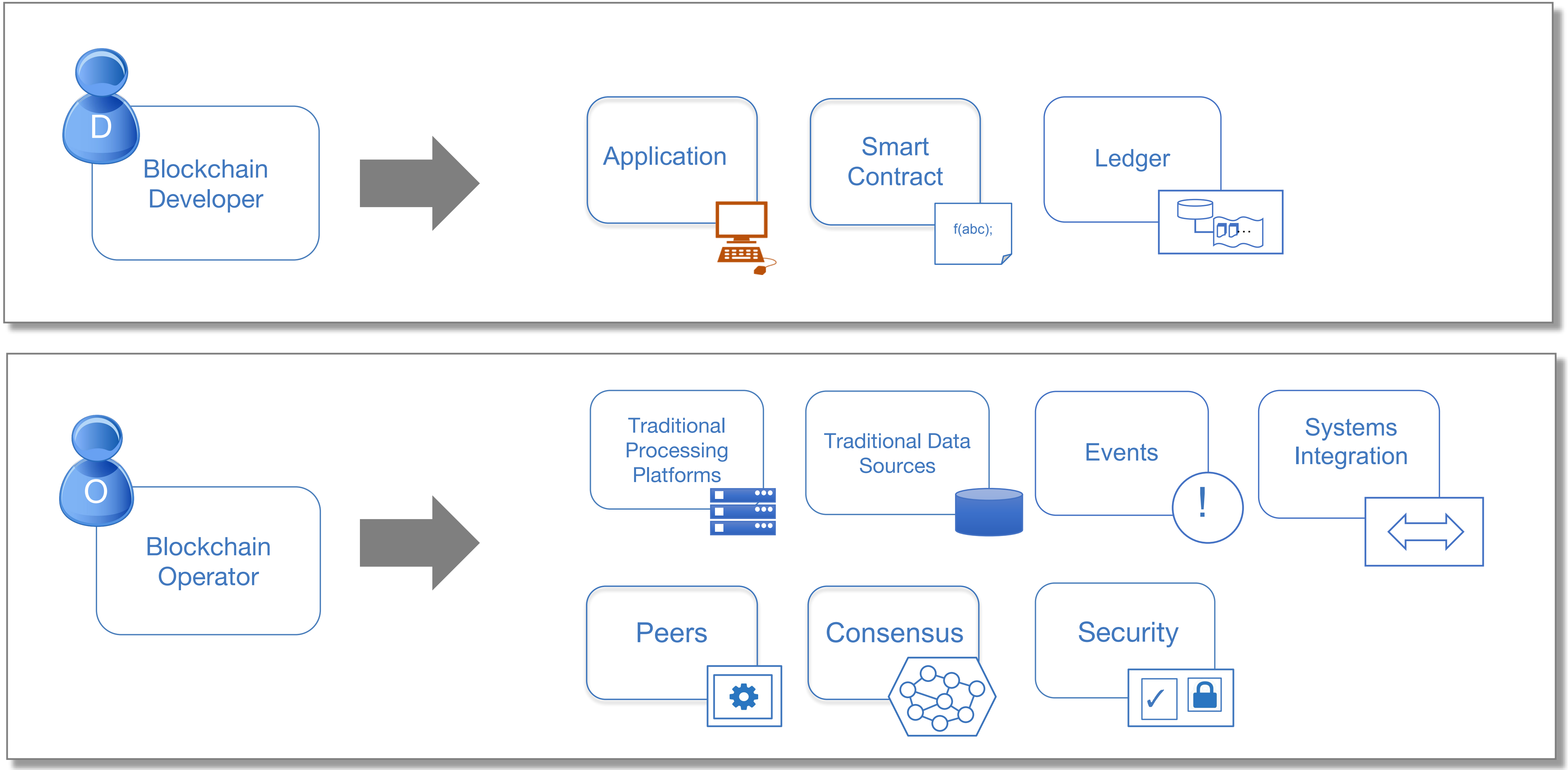
General

- ABN·AMRO
- AESTHETIC INTEGRATION
- ALTOROS
- ANZ
- 博图纵横 BOTUZONGHENG
- JM
- MonetaGo
- ML
- MOSCOW EXCHANGE
- MURPHY & MCGONIGLE A Professional Corporation
- NSE
- Orchestrating a brighter world NEC
- belink
- bitSE
- BLOCKCHAIN
- blocko
- Blockstream
- bloq
- NETKI
- NOKIA
- norbloc
- NTT DATA Global IT Innovator
- onchain
- 橙色魔方 Orange Magic Cube
- BNP PARIBAS
- BNY MELLON
- Broadridge
- bubi 布比
- ca technologies
- Calastone
- PAXOS
- PDX
- redhat
- Ribbit.me
- SAMSUNG SDS
- SANY
- CISCO
- cloudsoft
- CLS Fundamental to FX
- coinplug
- colu
- CONSENSYS
- SBERBANK
- GINGKOO 金丘股份 (837901)
- NEXGO
- 点融网 Dianrong.com
- Skry
- CREDITS
- Cuscal The complete payments partner
- ENERGY BLOCKCHAIN LABS
- Eurostep Digital
- FACTOM
- Gem
- SORAMITSU
- STATE STREET
- SWIFT
- swisscom
- symbiont
- TAI
- 众享比特 PeerSafe
- guardtime
- 33复杂美.CN
- HASHED HEALTH HEALTHCARE BLOCKCHAIN INNOVATION
- HUAWEI
- HUNDSUN
- tequa creek
- THOMSON REUTERS
- TMX
- UMP
- vmware
- WELLS FARGO
- 趣链科技 Hyperchain
- intellect
- intuit
- INVeSHARE
- IROOTECH
- KSD Korea Securities Depository
- Financial IT Partner koscom
- LedgerDomain
- Libra
- Lykke
- Milligan Partners
- MIRACL
- 云象 YUNPHANT
- 梧桐树 wutongtree.com
- 保全网 BaoQuan.com

Associate

- CHAMBER OF DIGITAL COMMERCE
- CSA cloud security alliance
- Investrata Foundation
- NXT FOUNDATION
- sovrin
- fondazione INUIT Università di Roma Tor Vergata

Blockchain key actors and their domains



Blockchain Developer

Application

Smart Contract

Ledger

Blockchain Operator

Traditional Processing Platforms

Traditional Data Sources

Events

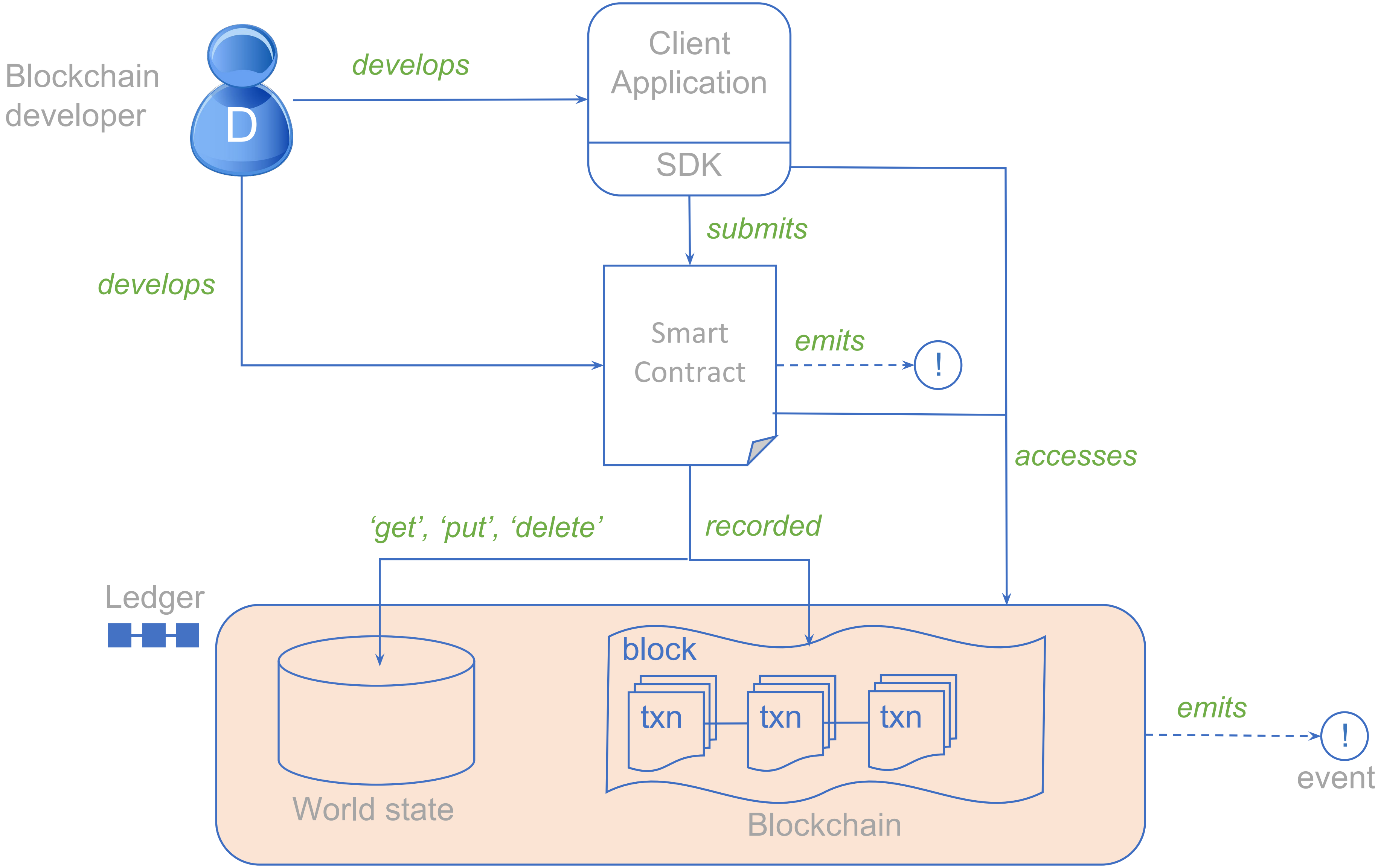
Systems Integration

Peers

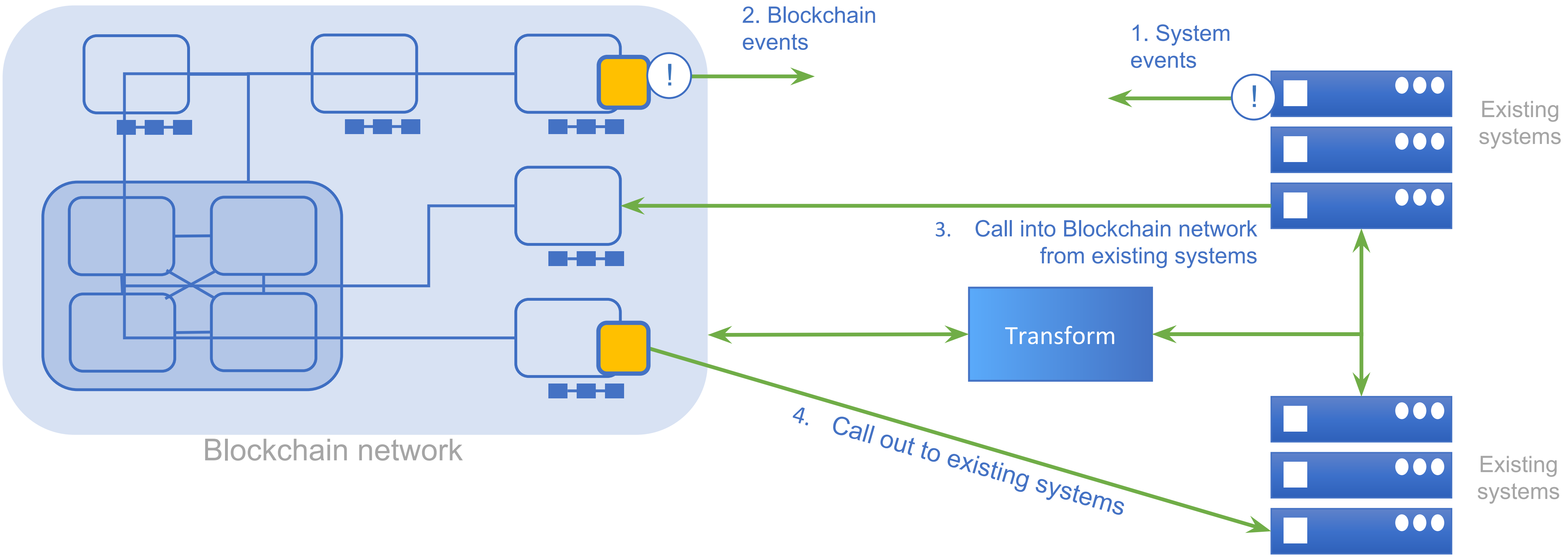
Consensus

Security

How applications interact with the ledger

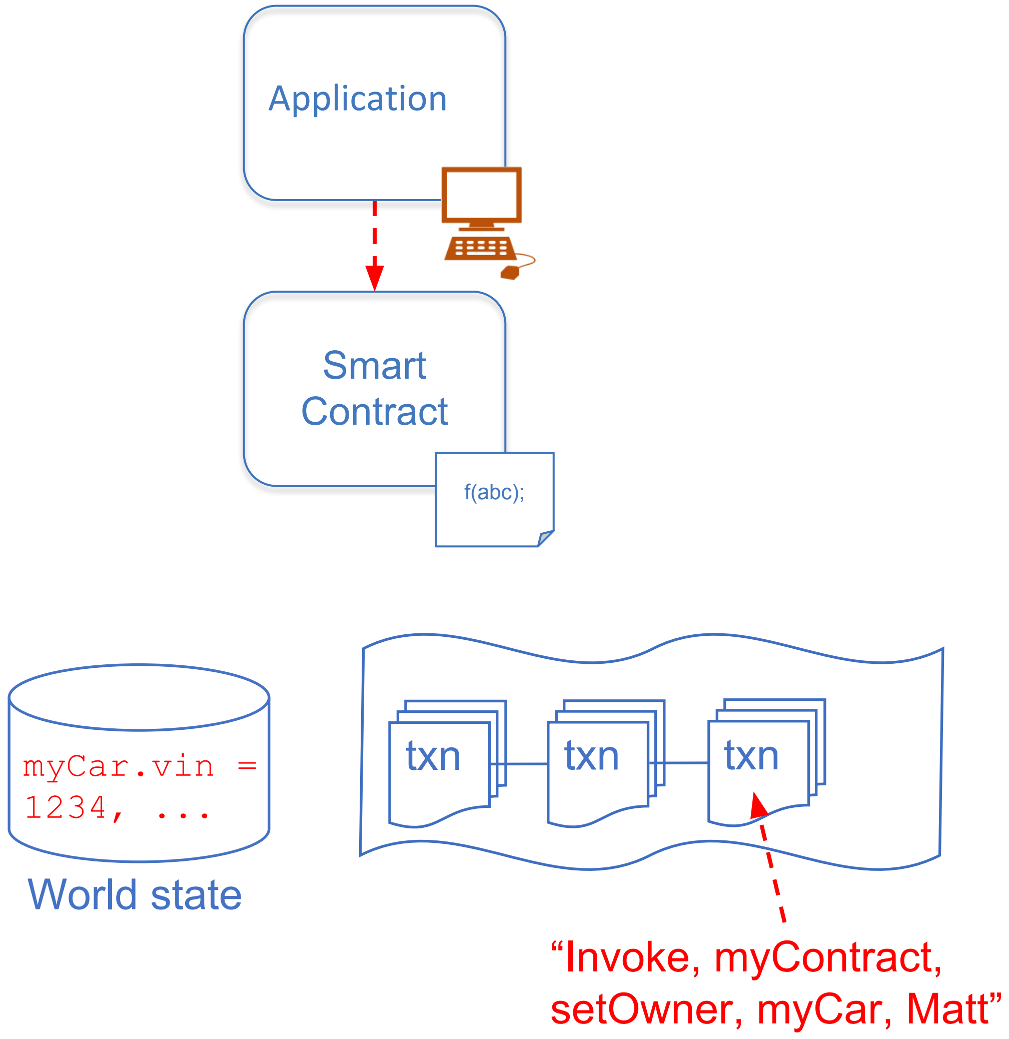


Integrating with existing systems – possibilities



Working with the ledger:

Example of a change of ownership transaction



Transaction input - sent from application

```
invoke(myContract, setOwner,
          myCar, Matt)
…
```

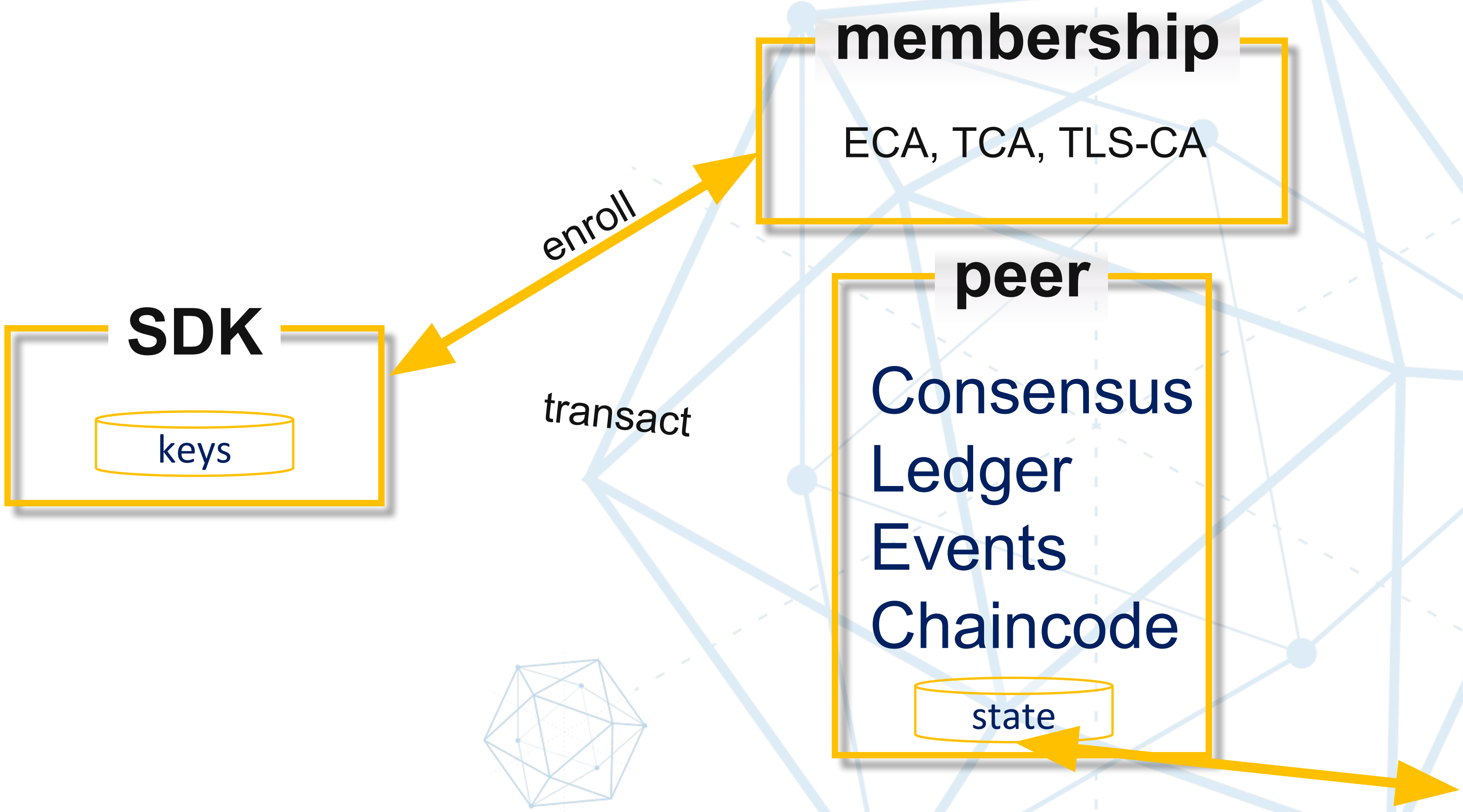
Smart contract implementation

```
setOwner(Car, newOwner) {
   set Car.owner = newOwner
}
```

World state: new contents

```
myCar.vin = 1234
myCar.owner = Matt
myCar.make = Audi
…
```

Architecture of Hyperledger Fabric v0.6



Overview of Hyperledger Fabric v1 – Lessons Learned

- Better reflect business processes by specifying who endorses transactions
- Support broader regulatory requirements for privacy and confidentiality
- Scale the number of participants and transaction throughput
- Eliminate non deterministic transactions
- Support rich data queries of the ledger
- Dynamically upgrade fabric and chaincode
- Support for multiple credential and cryptographic services for identity
- Support for ”bring your own identity”

Endorsement, Ordering and Validation

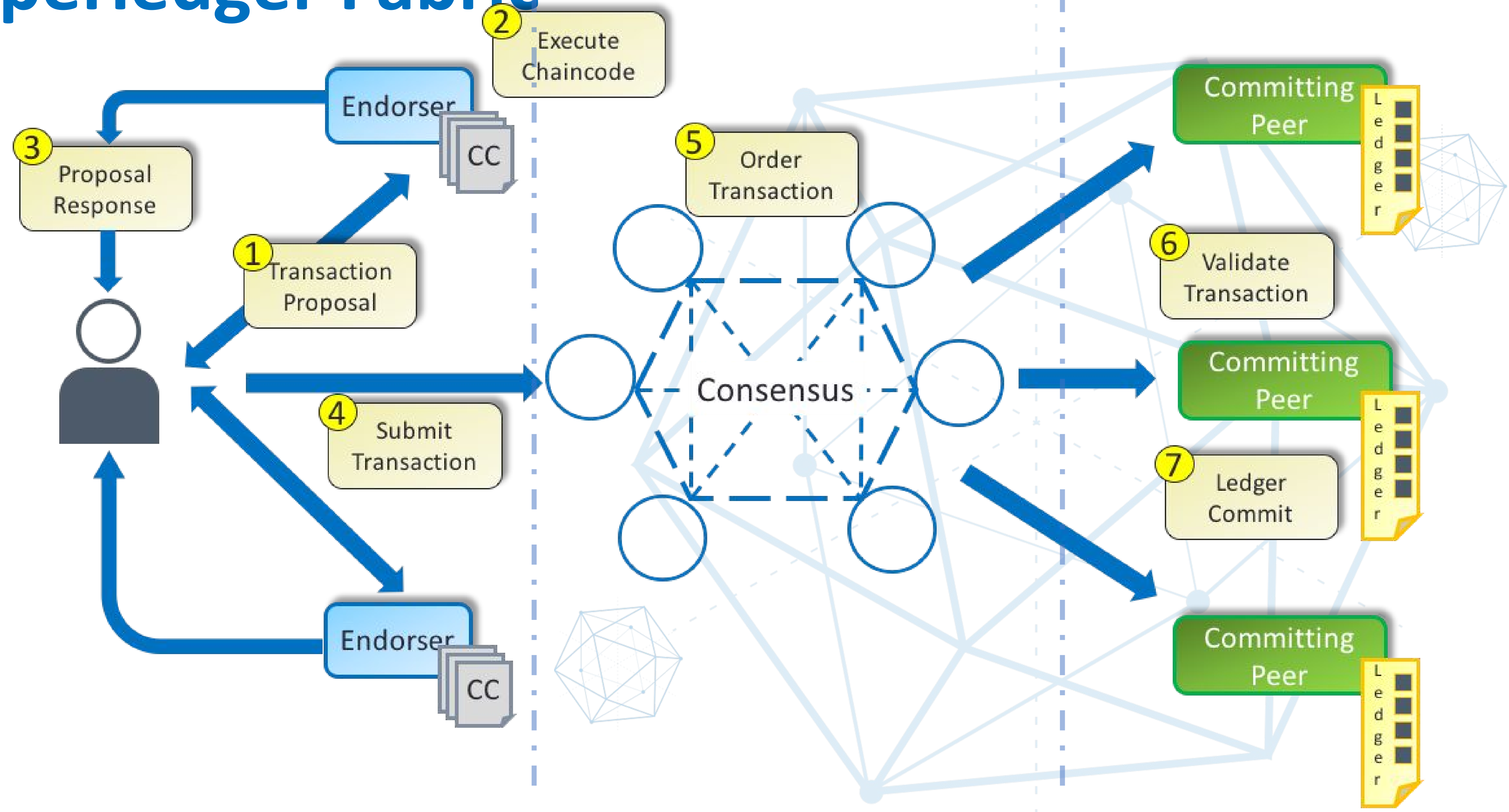
Nodes and roles

⬜ **Committing Peer**: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).

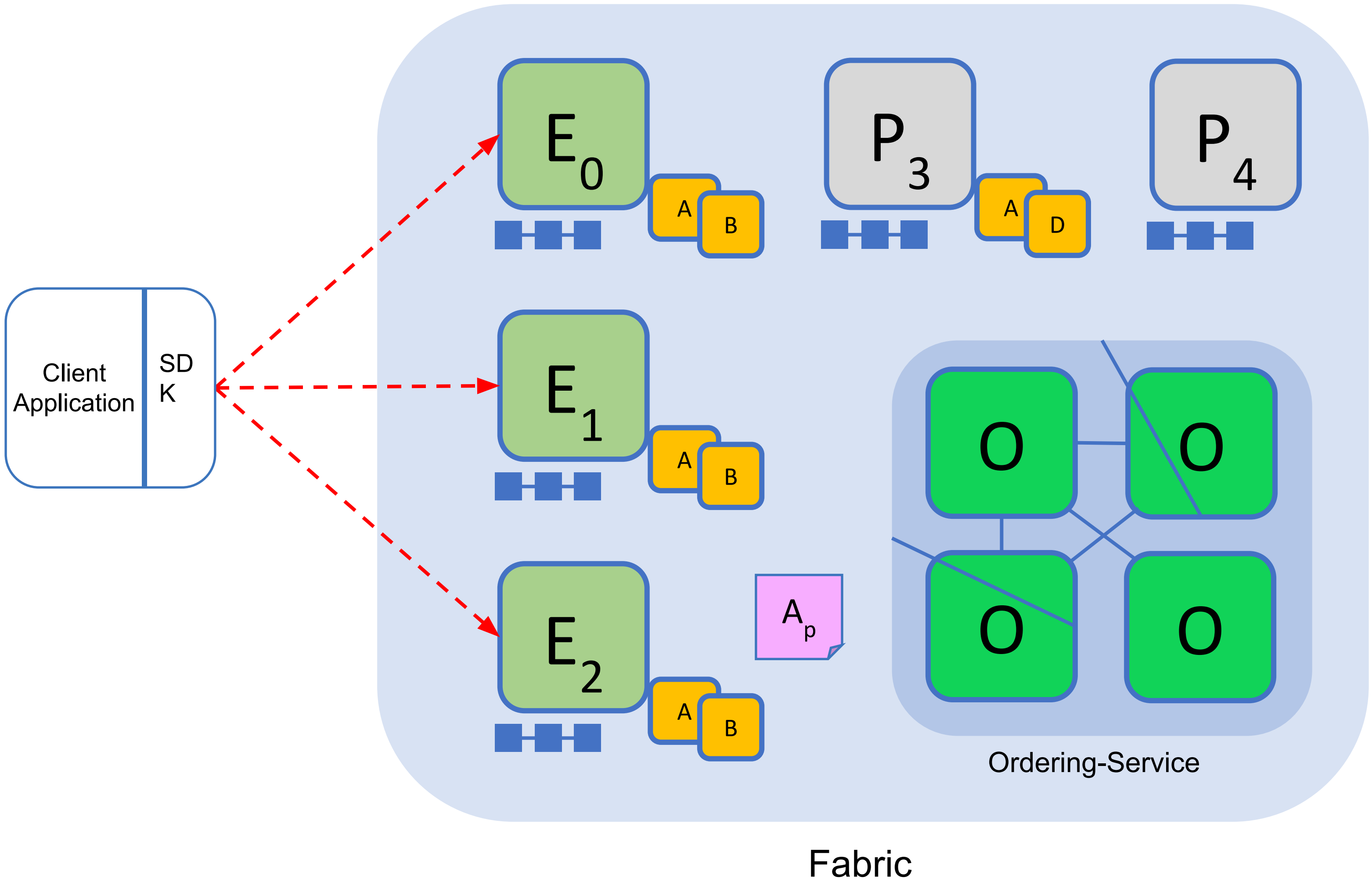
🟩 **Endorsing Peer**: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract

🟩 **Ordering Nodes (service)**: Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

Hyperledger Fabric



Sample transaction: Step 1/7 – Propose transaction



Application proposes transaction

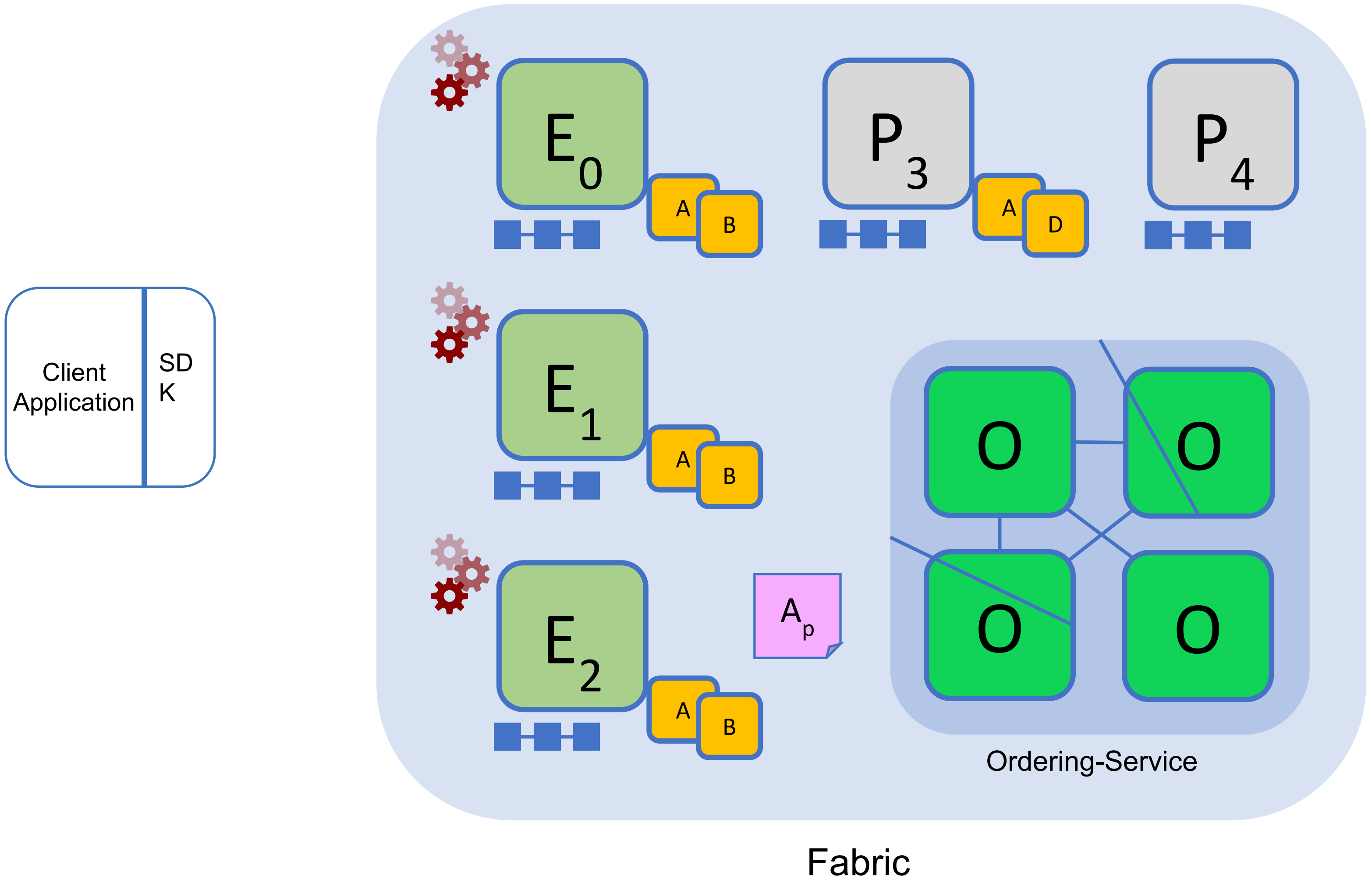
Endorsement policy:
- "$E_{0,}$ E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E_0, E_1, E_2}

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Sample transaction: Step 2/7 – Execute proposal



Endorsers Execute Proposals

E_0, E_1 & E_2 will each execute the *proposed* transaction. None of these executions will update the ledger

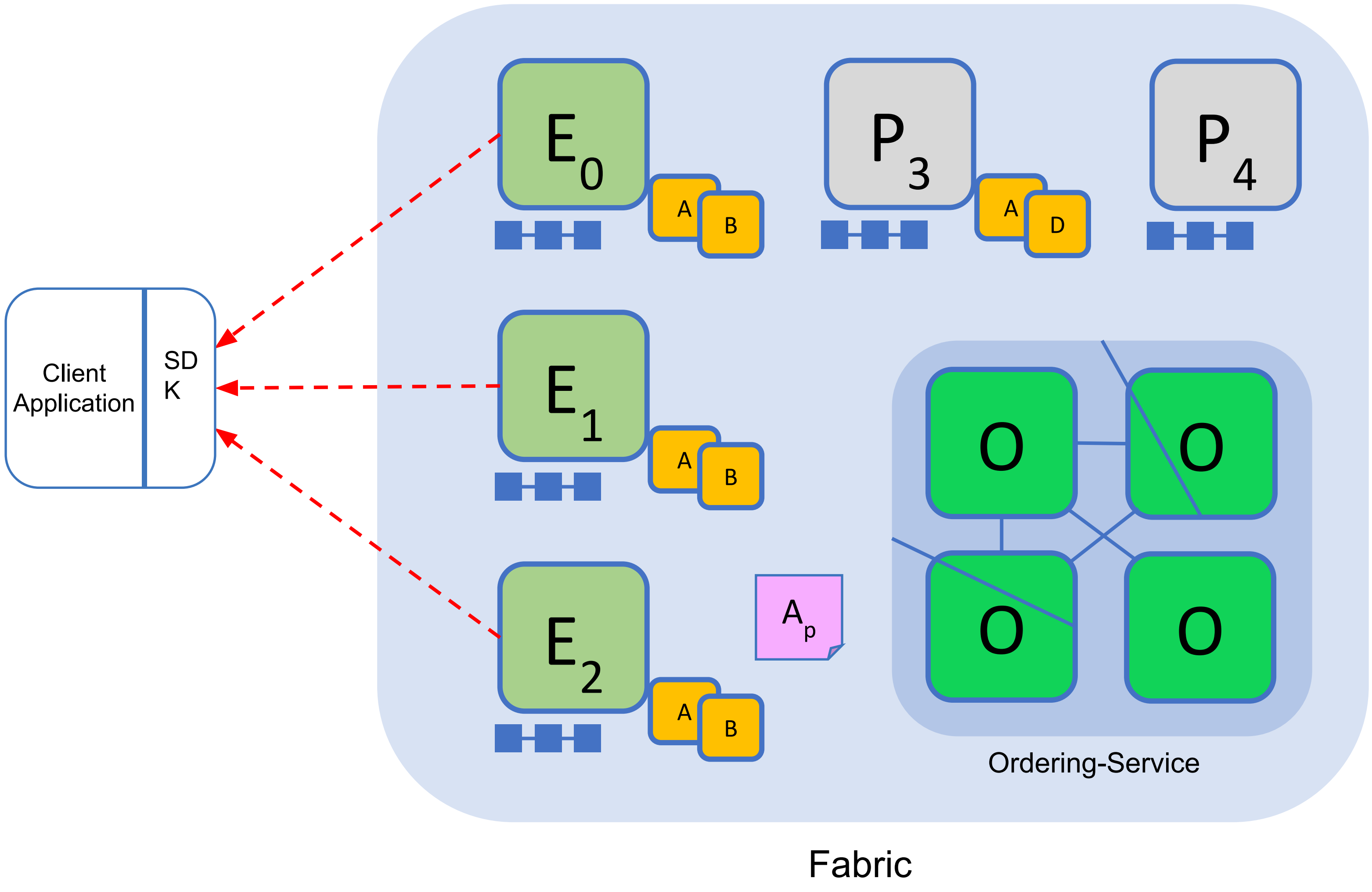
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Sample transaction: Step 3/7 – Proposal Response



Application receives responses

RW sets are asynchronously returned to application

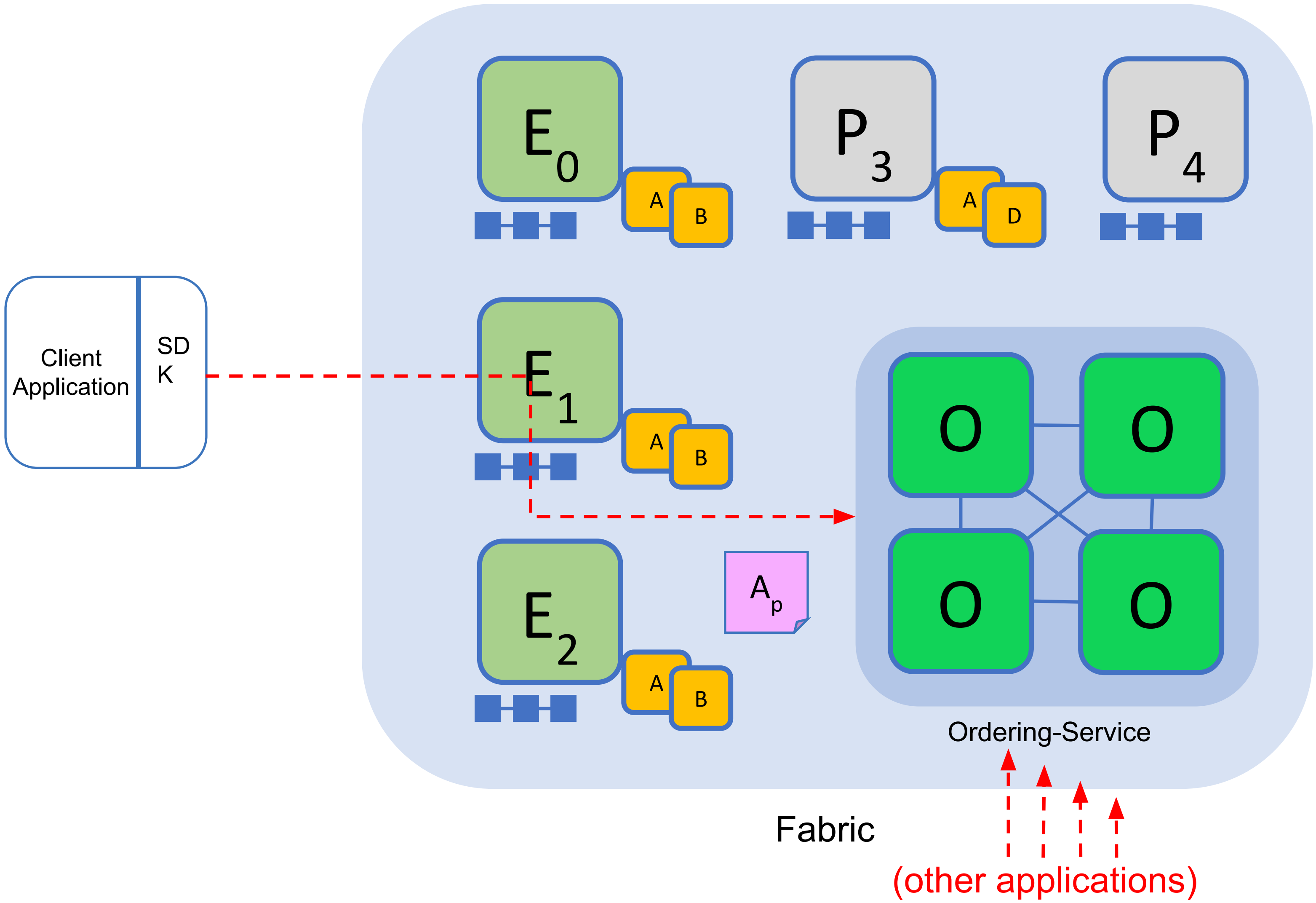
The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

Key:

Endorser	🟩	▬	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Sample transaction: Step 4/7 – Order Transaction



Application submits responses for ordering

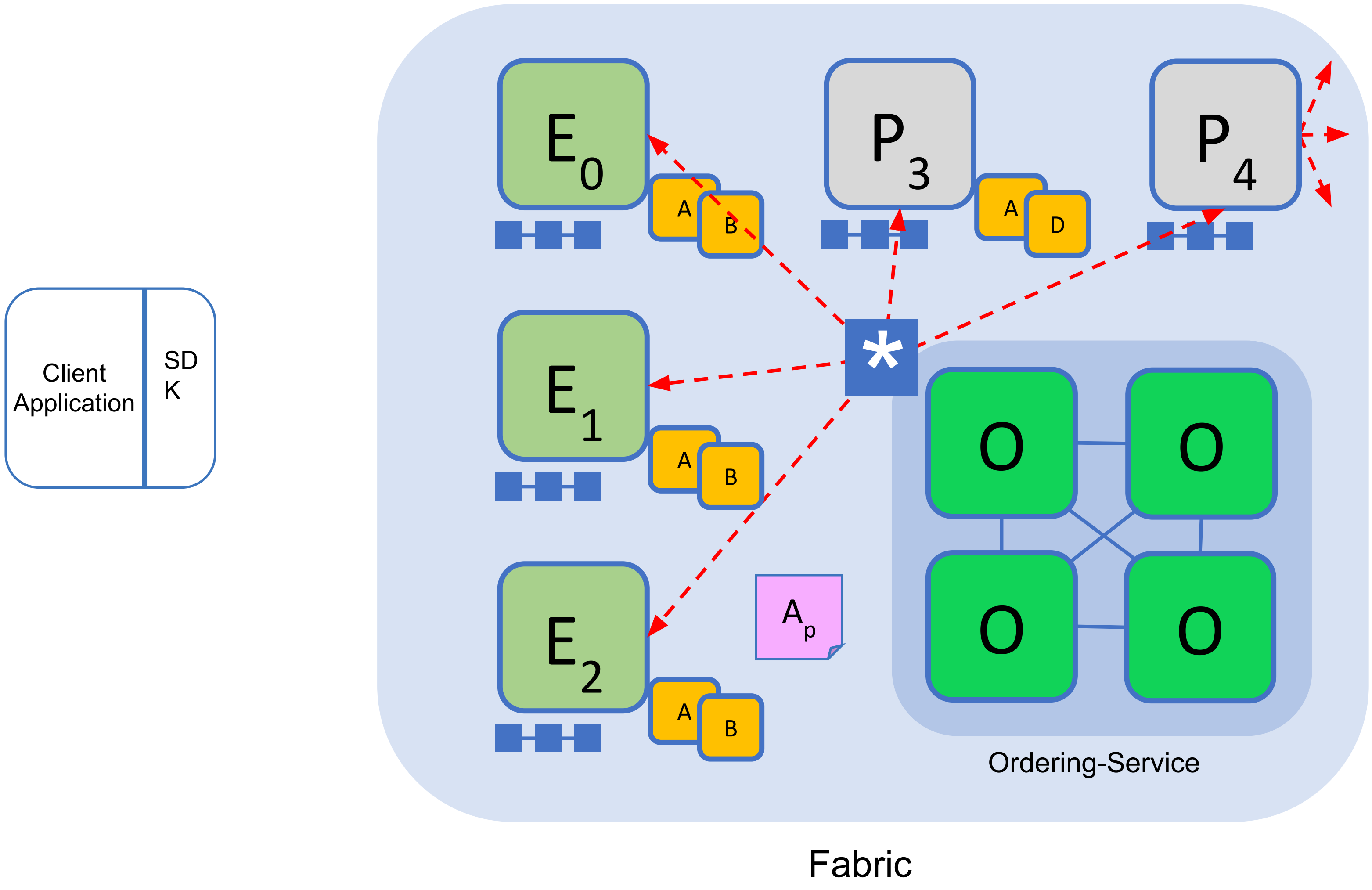
Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	▨	Endorsement Policy

Sample transaction: Step 5/7 – Deliver Transaction



Orderer delivers to all committing peers

Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

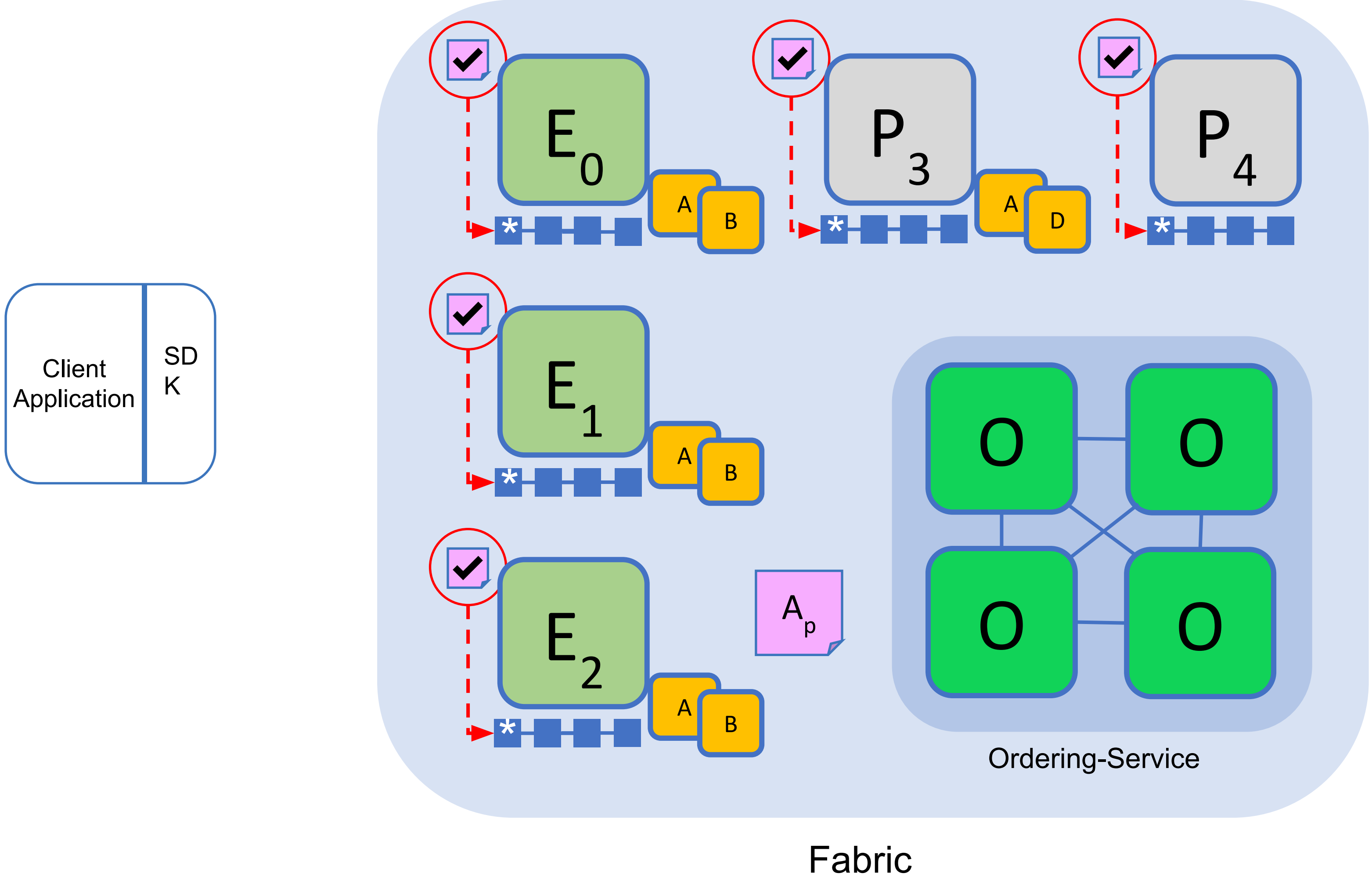
Different ordering algorithms available:

- SOLO (Single node, development)
- Kafka (Crash fault tolerance)
- SBFT (Byzantine fault tolerance)

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Sample transaction: Step 6/7 – Validate Transaction



Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

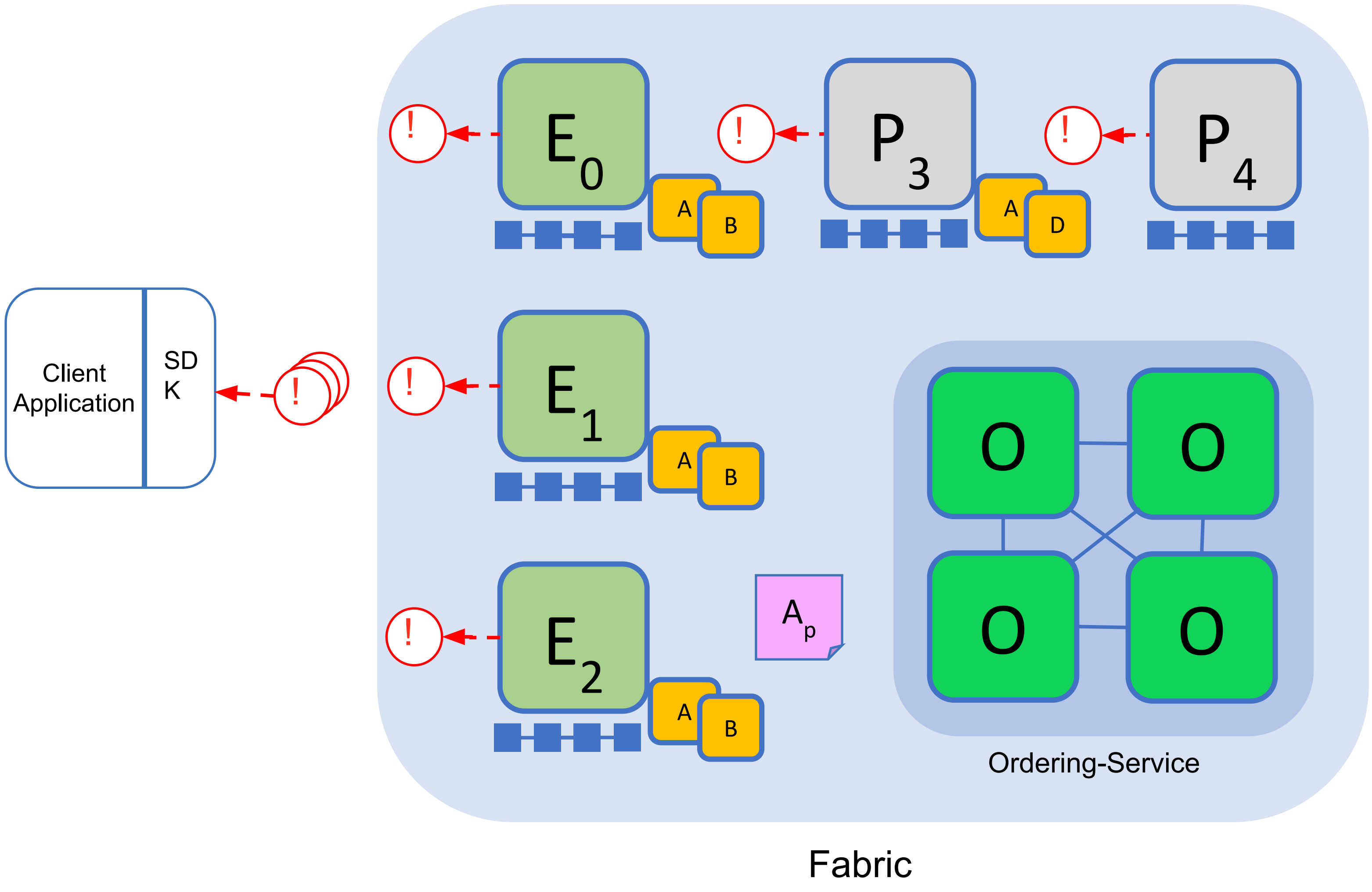
Validated transactions are applied to the world state and retained on the ledger

Invalid transactions are also retained on the ledger but do not update world state

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Sample transaction: Step 7/7 – Notify Transaction



Committing peers notify applications

Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

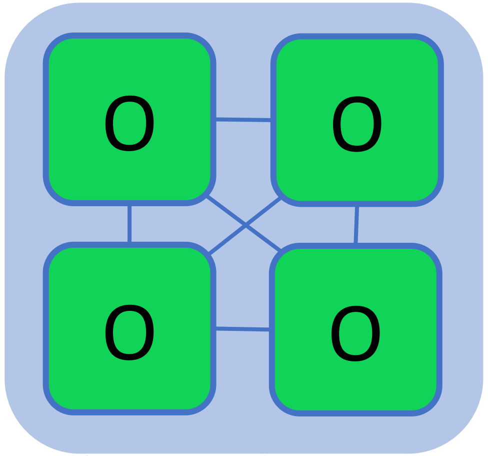
Applications will be notified by each peer to which they are connected

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Ordering Service

The ordering service packages transactions into blocks to be delivered to peers. Communication with the service is via channels.



Ordering-Service

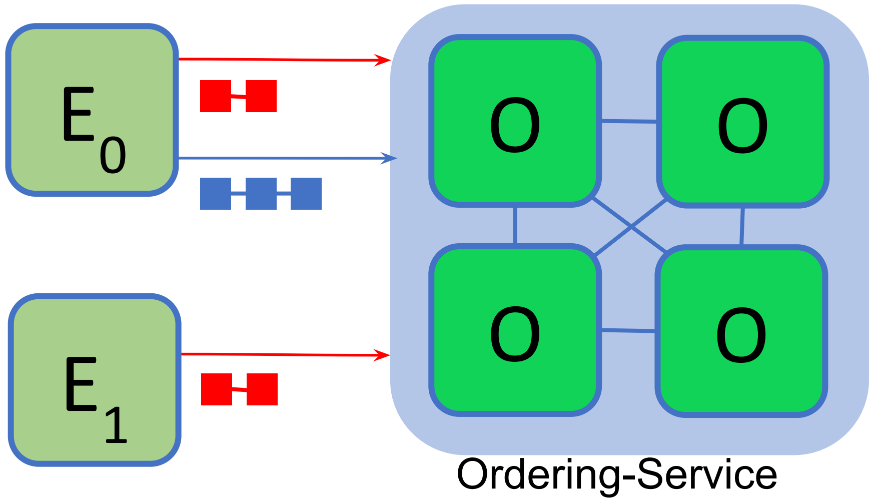
Different configuration options for the ordering service include:

- **SOLO**
  - Single node for development
- **Kafka :** Crash fault tolerant consensus
  - 3:n nodes minimum
  - Odd number of nodes recommended
- **SBFT :** Byzantine fault tolerant consensus
  - 4:n nodes minimum

Channels

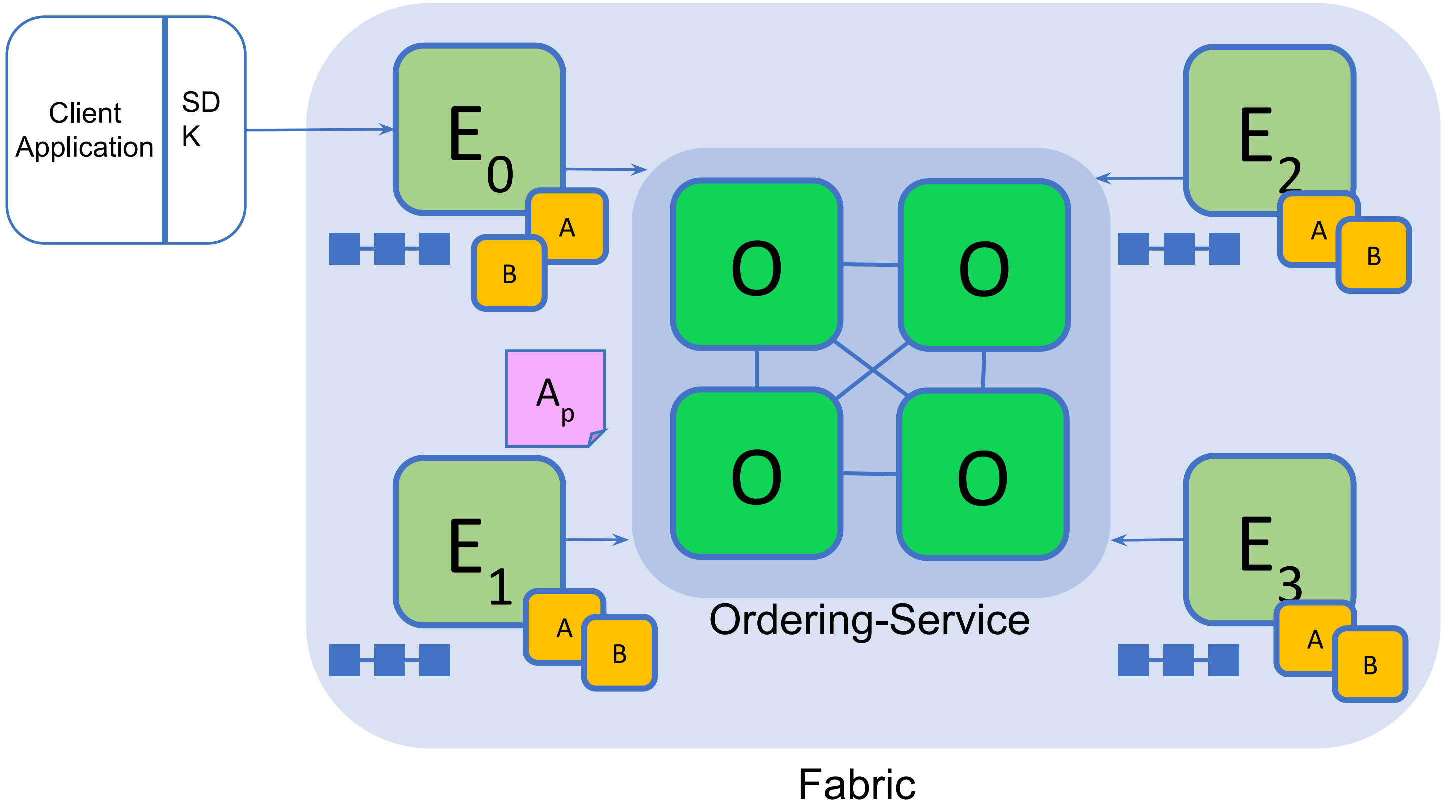
Channels

Separate channels isolate transactions on different ledgers



- Chaincode is installed on peers that need to execute business logic and participate in endorsement process
- Chaincode is instantiated on specific channels for specific peers
- Ledgers exist in the scope of a channel
  - Ledgers can be shared across an entire network of peers
  - Ledgers can be included only on a specific set of participants
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

Single Channel Endorsement

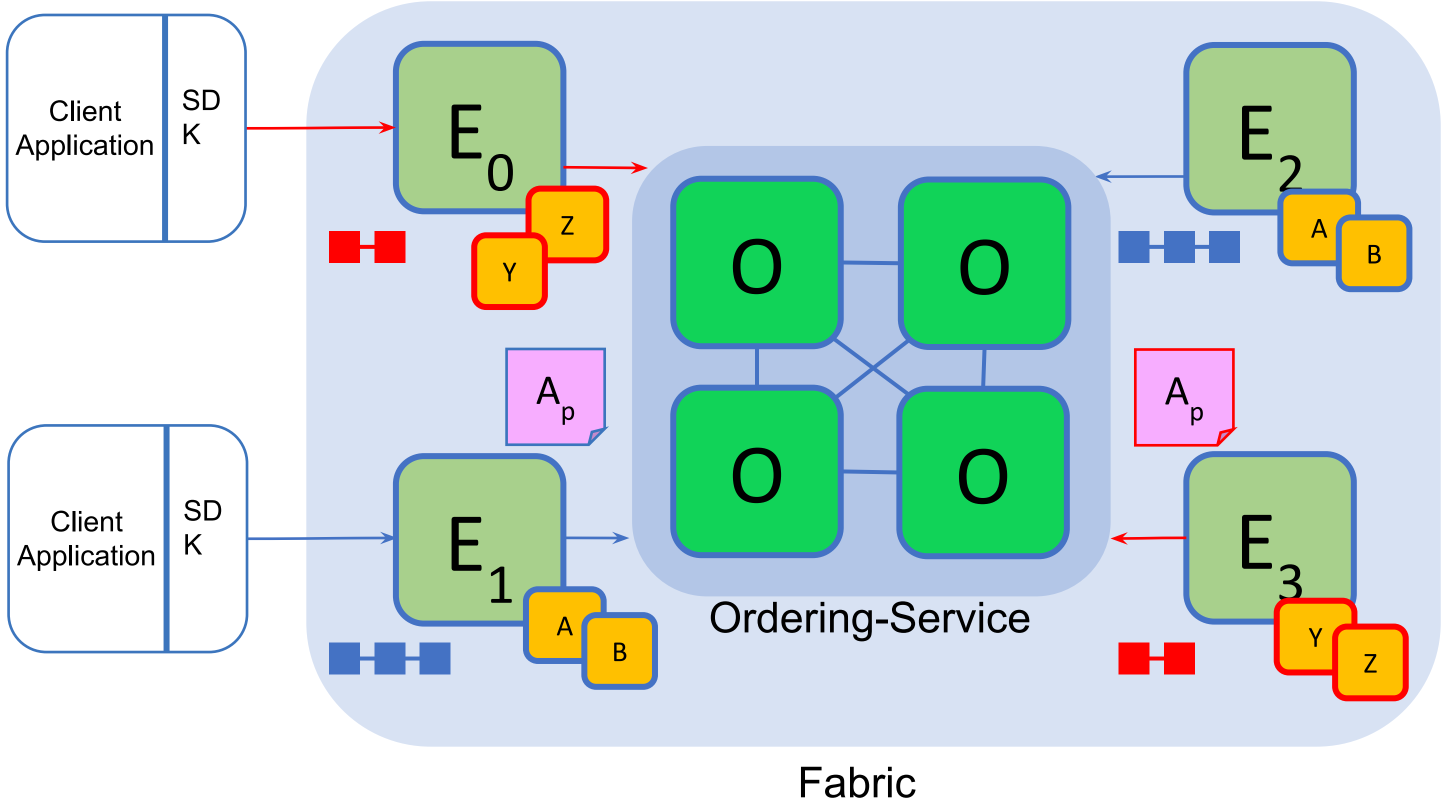


- All peers connect to the same channel (blue).
- All peers consider the same chaincodes for execution and maintain the same ledger
- Endorsement by peers E_{0}, E_{1}, E_{2} and E_3

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟨	🟪	Endorsement Policy

Multi Channel & Chaincode Endorsement



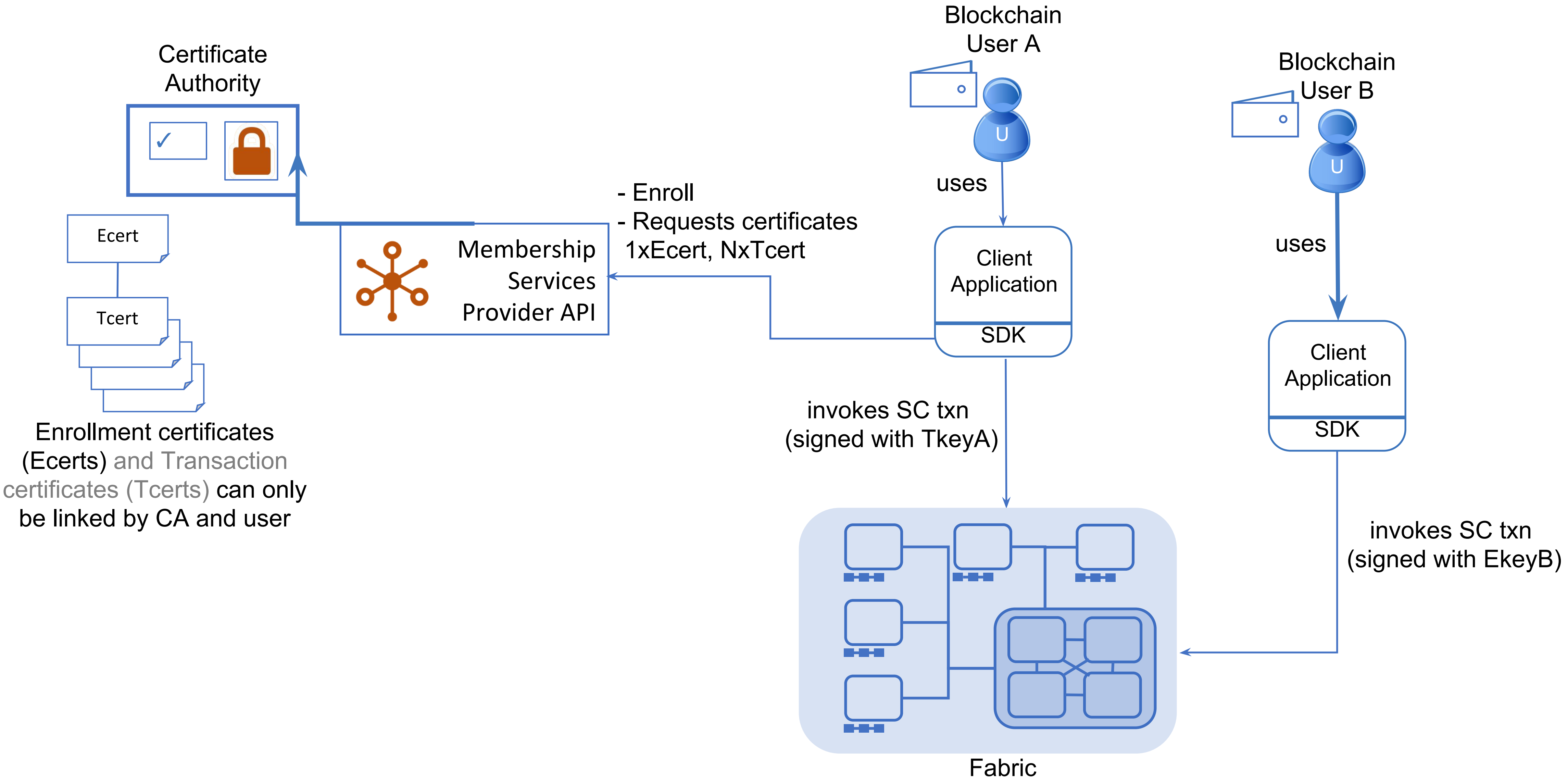
- Peers E_0 and E_3 connect to the red channel for chaincodes Y and Z
- Peers E_1 and E_2 connect to the blue channel for chaincodes A and B

Key:

Endorser	🟩	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chain code)	🟧	▨	Endorsement Policy

Identity Management

Membership Services Overview



Certificate Authority

Ecert

Tcert

Enrollment certificates (Ecerts) and Transaction certificates (Tcerts) can only be linked by CA and user

Membership Services Provider API

- Enroll
- Requests certificates 1xEcert, NxTcert

Blockchain User A

uses

Client Application

SDK

invokes SC txn (signed with TkeyA)

Blockchain User B

uses

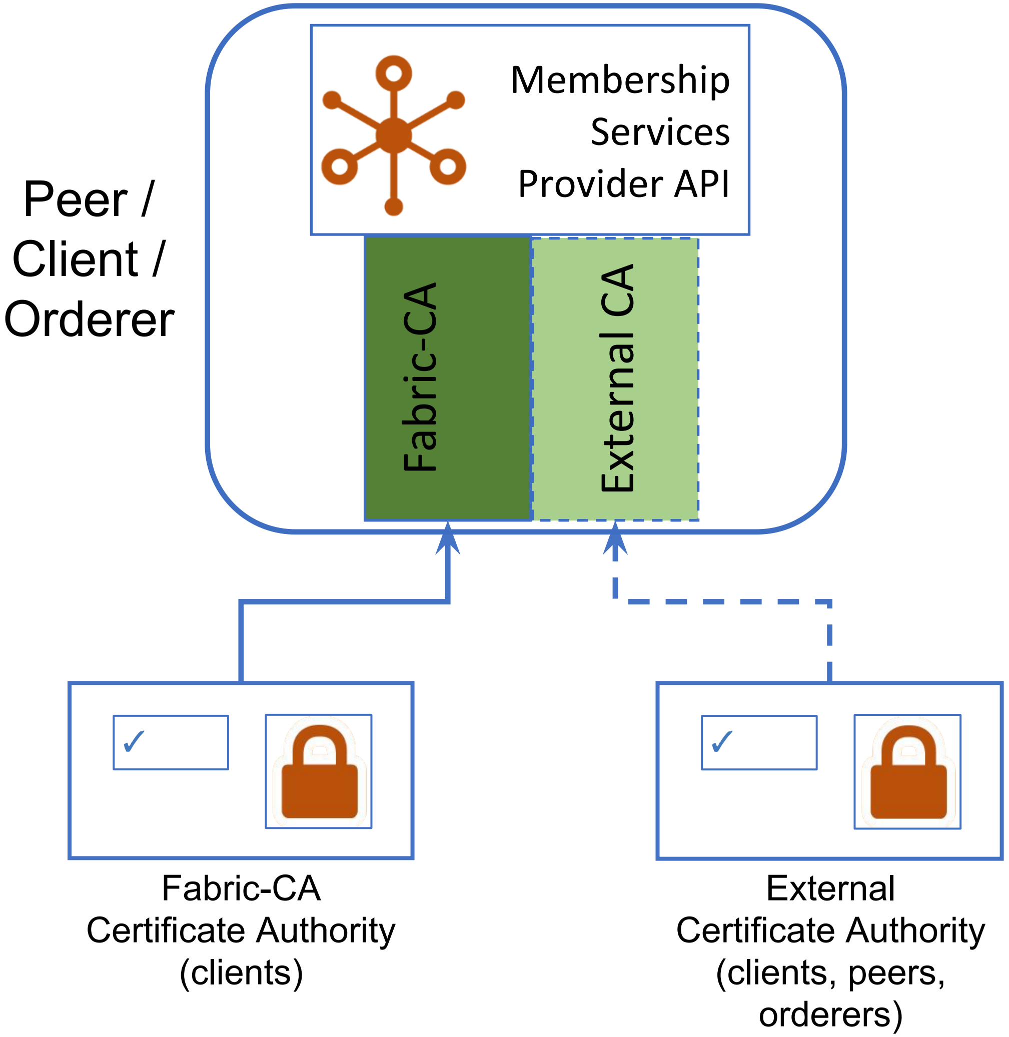
Client Application

SDK

invokes SC txn (signed with EkeyB)

Fabric

Membership Services Provider API

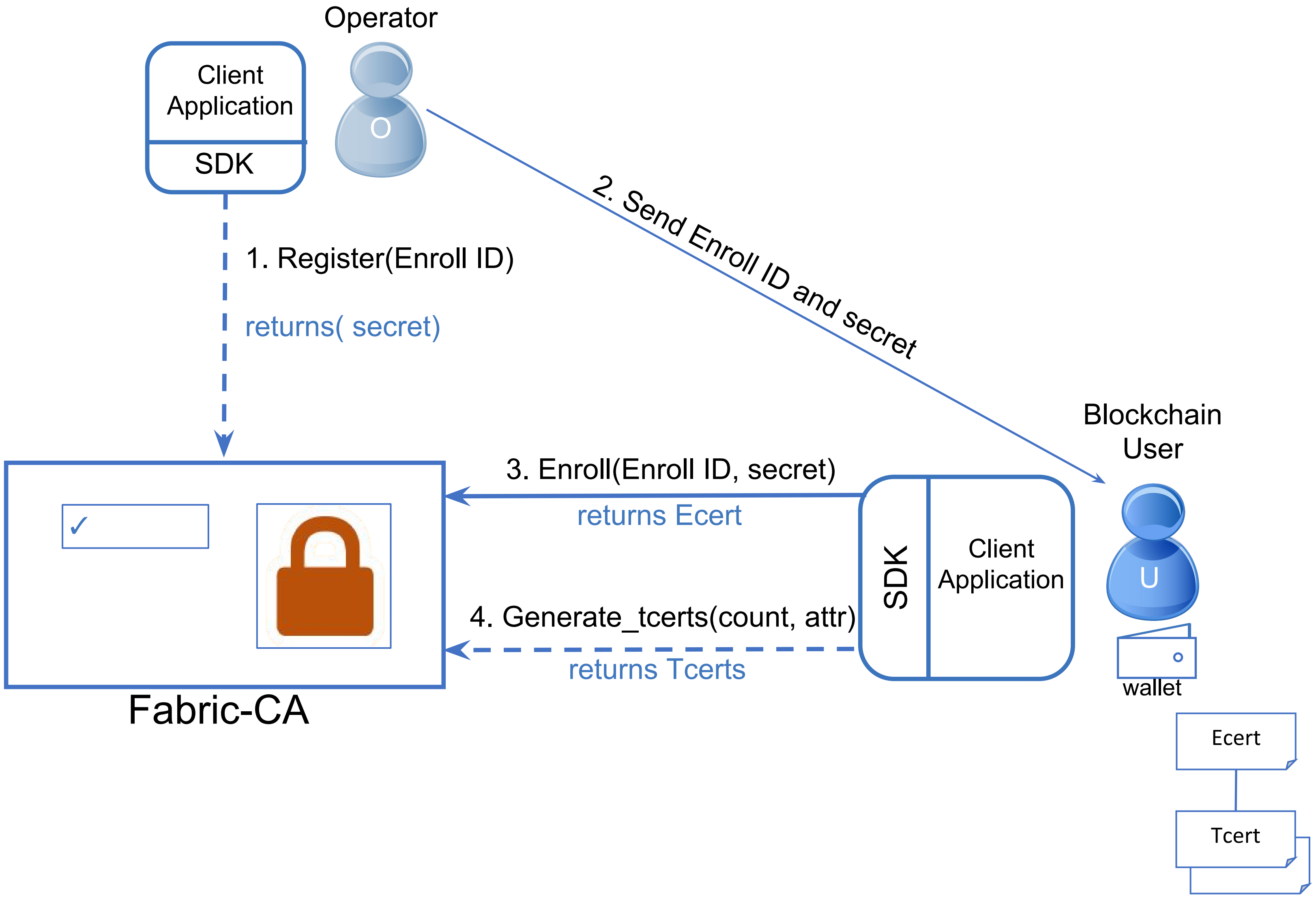


- Pluggable interface supporting a range of credential architectures

- Governs identity for Peers, Users, and Orderers.

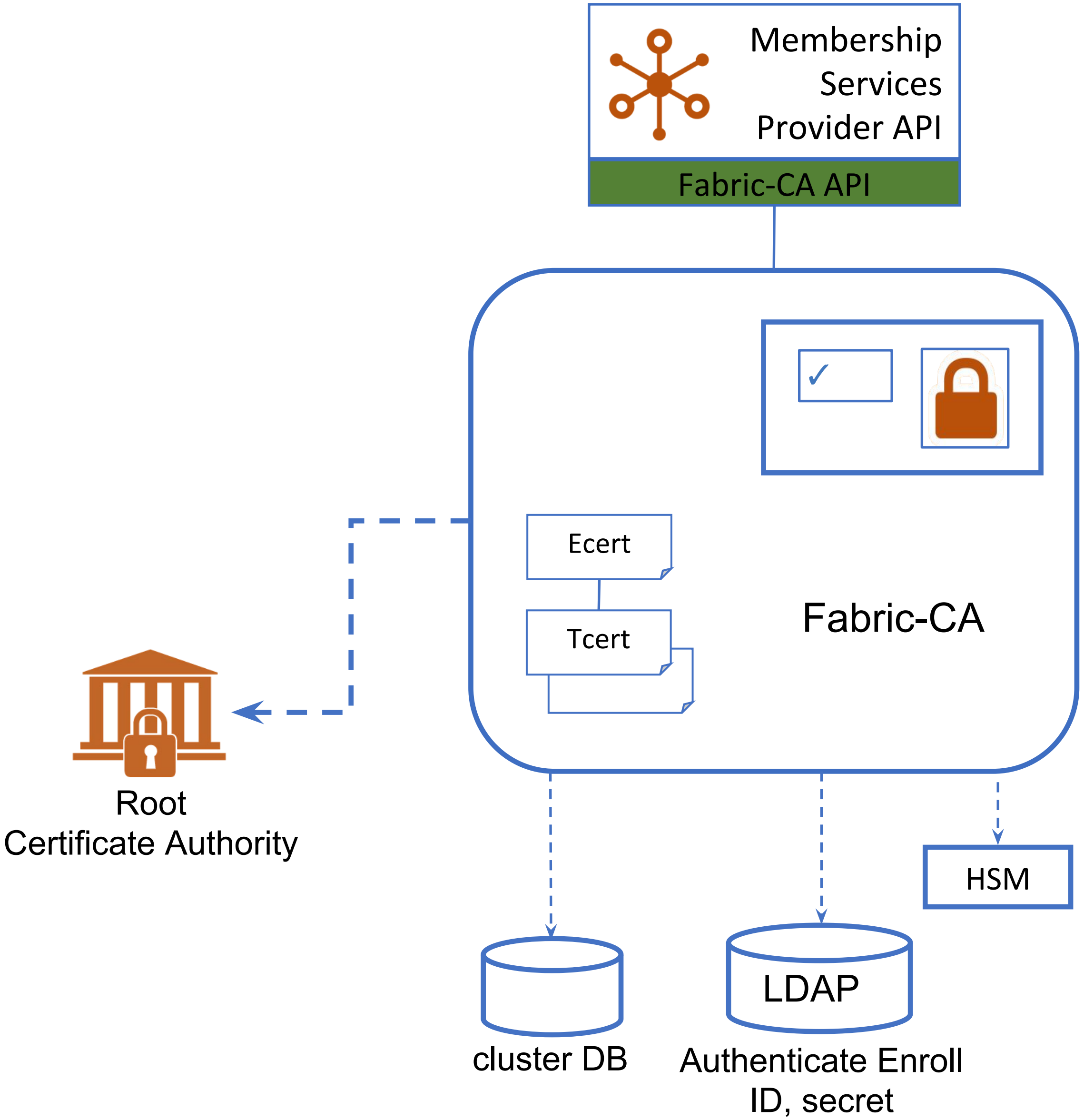
- Provides:
    - Concrete identity format
    - User credential validation
    - User credential revocation
    - Signature generation and verification
    - (Optional) credential issuance

Fabric-ca: New User Registration and Enrollment



- Admin registers new user with Enroll ID
- User enrolls and receives credentials
- User requests Tcerts in batches
- Additional offline registration and enrollment options available

Fabric-CA Details


Membership Services Provider API
Fabric-CA API
✓
Ecert
Tcert
Fabric-CA
Root Certificate Authority
HSM
cluster DB
LDAP
Authenticate Enroll ID, secret

- Default implementation of the Membership Services Provider Interface to cover identity management
- Issues Ecerts (long-term identity) and Tcerts (disposable certificate)
- Supports clustering for HA characteristics
- Supports LDAP for user authentication
- Supports HSM

Transaction and Identity Privacy

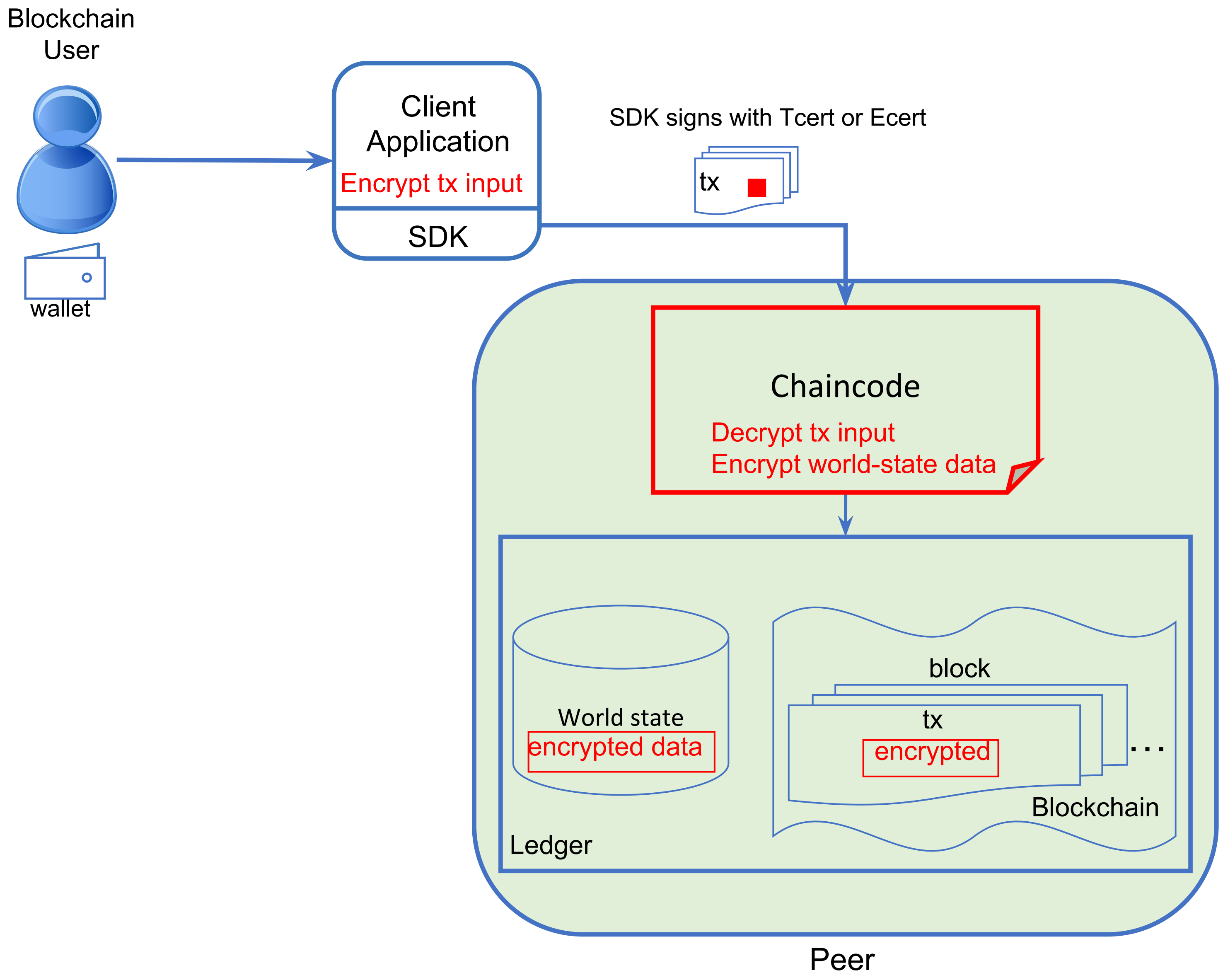
- Enrollment Certificates, Ecerts
  - Long term identity
  - Can be obtained offline, bring-your-own-identity

- [Transaction Certificates, Tcerts]
  - Disposable certificates, typically used once, requested from Transaction CA
  - Tcert derived from long term identity - Enrollment Certificate, Ecert
  - Only Transaction CA can link Ecert and Tcert

- Permissioned Interactions
  - Users sign with either Ecerts or Tcerts

- Membership Services
  - Abstract layer to credential providers

Application Level Encryption



Handled in the application domain.

Multiple options for encrypting:

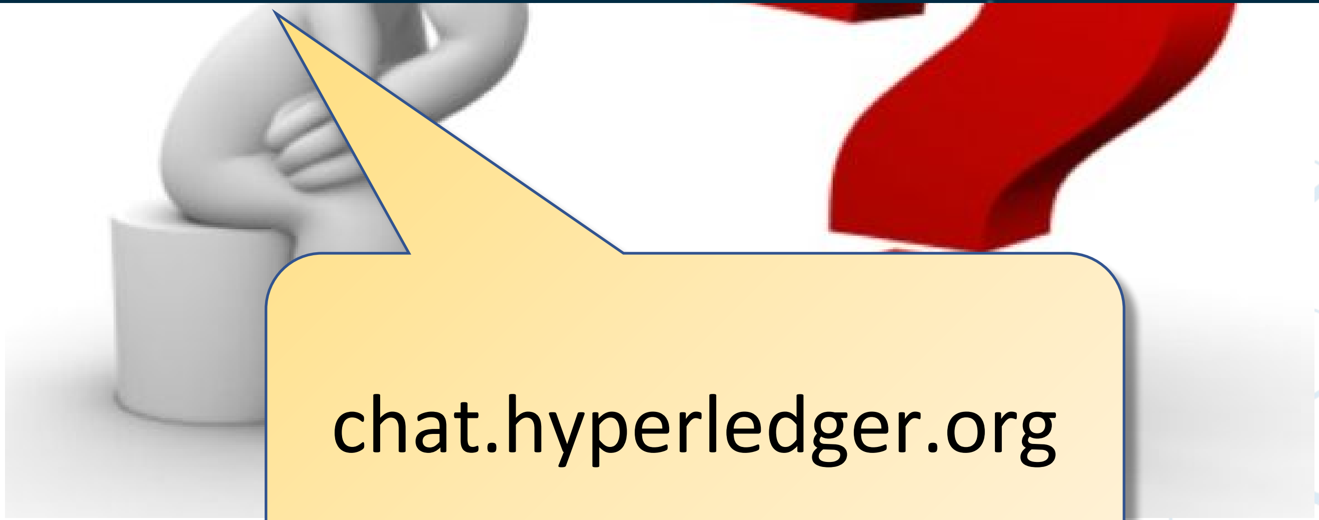
- Transaction Data
- Chaincode*
- World-State data

Chaincode optionally deployed with cryptographic material, or receive it in the transaction from the client application using the *transient* data field (not stored on the ledger).

*Encryption of application chaincode requires additional development of system chaincode.

QUESTIONS?

ROCKET.CHAT



chat.hyperledger.org

Thank you!