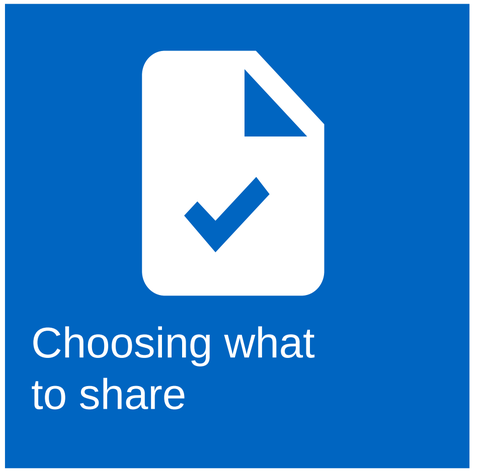
Hyperledger Fabric 的数据隐私保护

赵振华

Requirements of blockchain for business

Participants know who they are dealing with



Participants decide which assets to share

Participants give provable endorsement

Information shared via need-to-know

Membership Service Provider

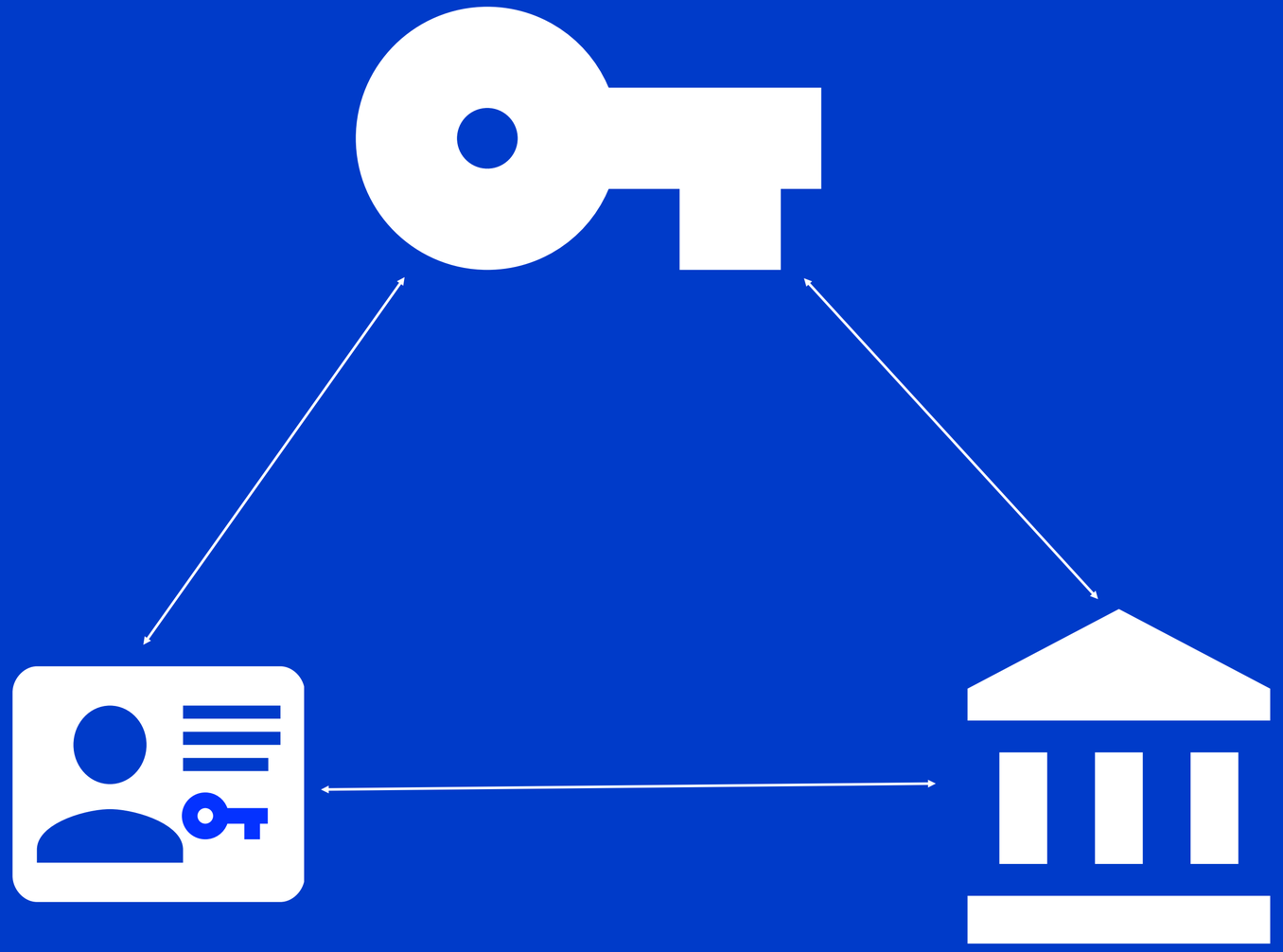
Multichannel Consensus

Private Data

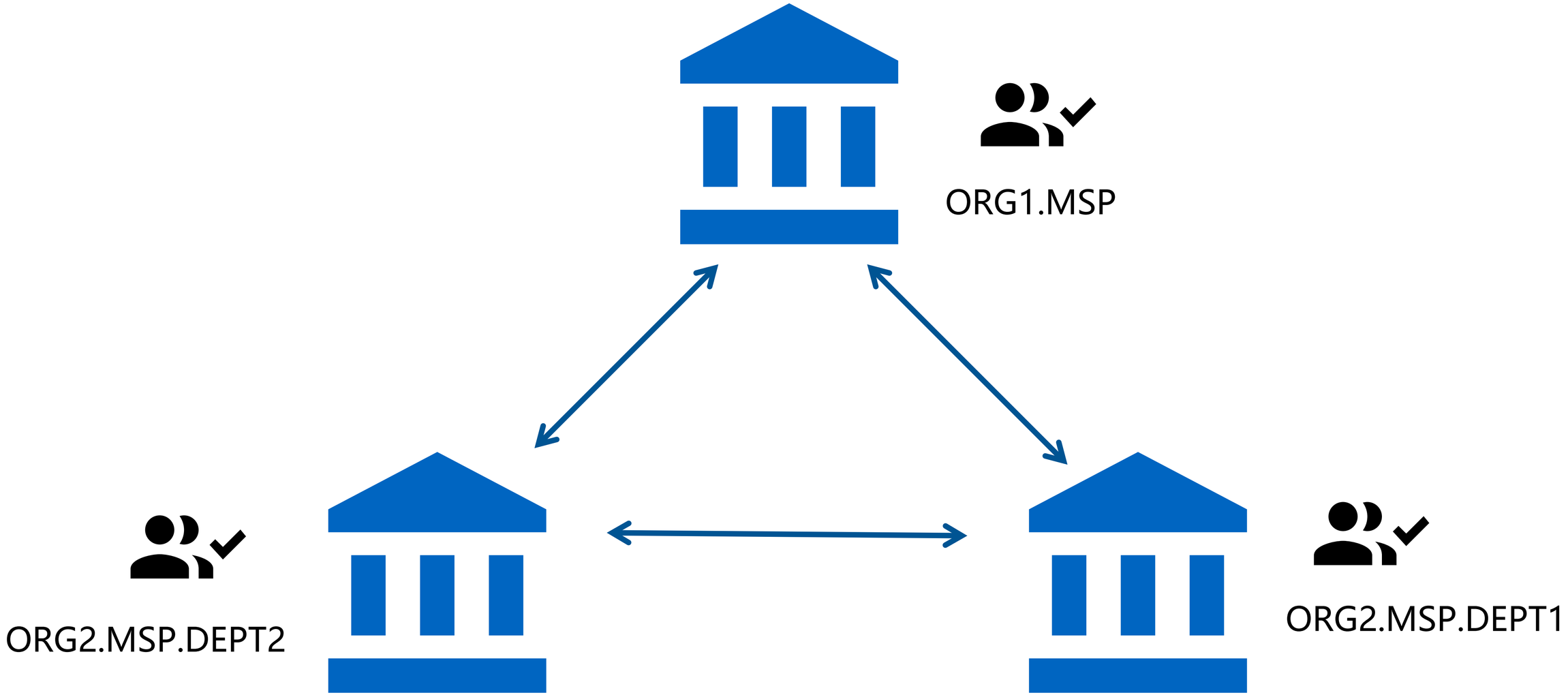
Membership Service Provider

- 企业业务决定
  - 业务隐私性——只和特定组织发生交易
  - 安全——数据泄露给企业带来损失
- 法律法规
  - KYC, AML（反洗钱法）
  - 食品安全管理条例
- 技术实现
  - 加密、签名

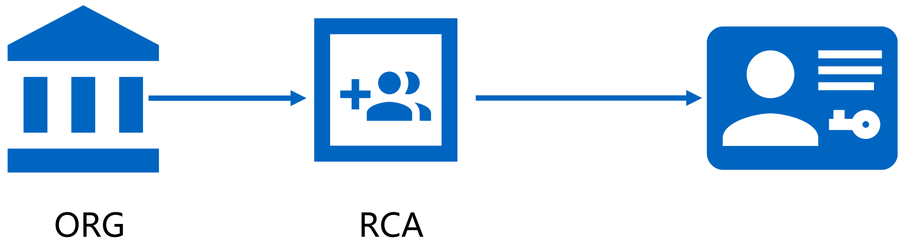
MSP 组成

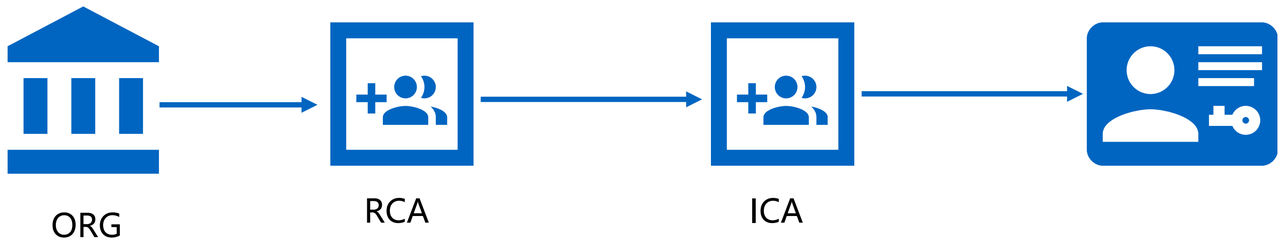


MSP & Org Mapping

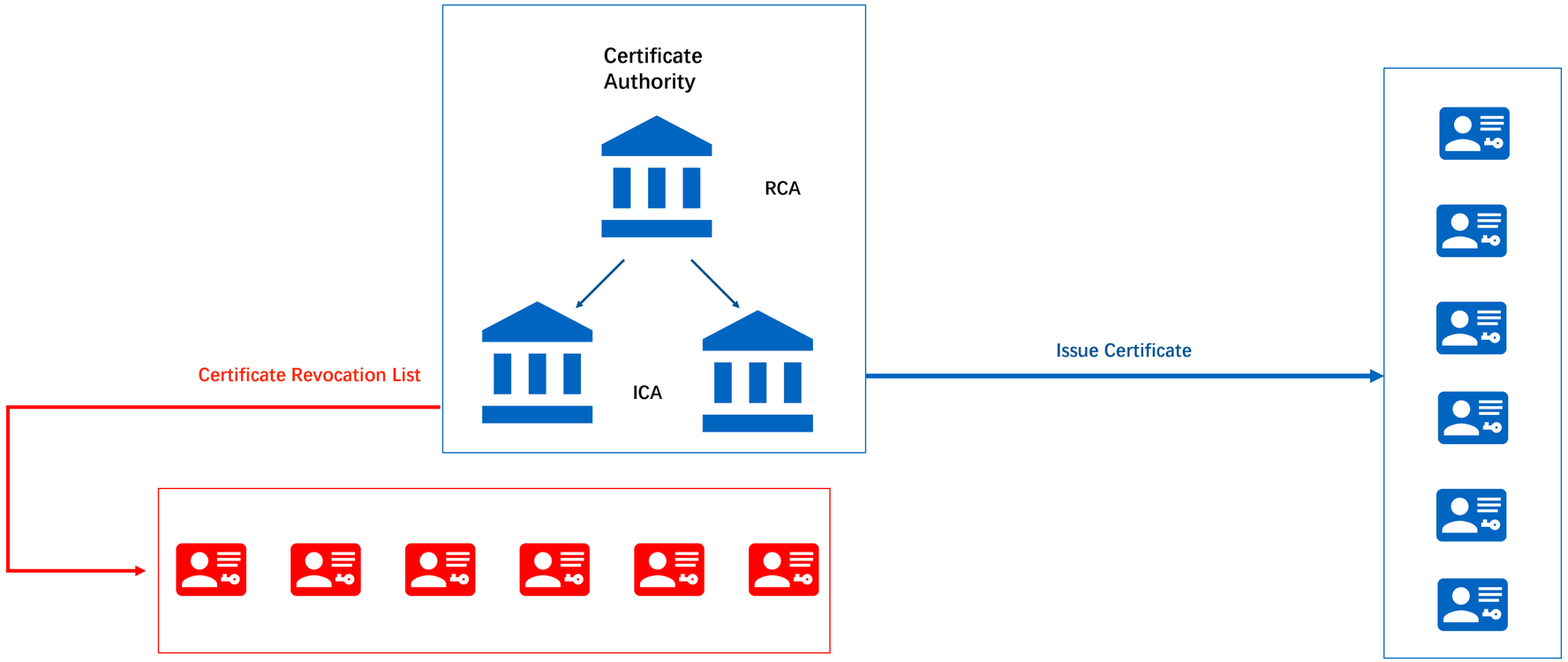


RCA & ICA

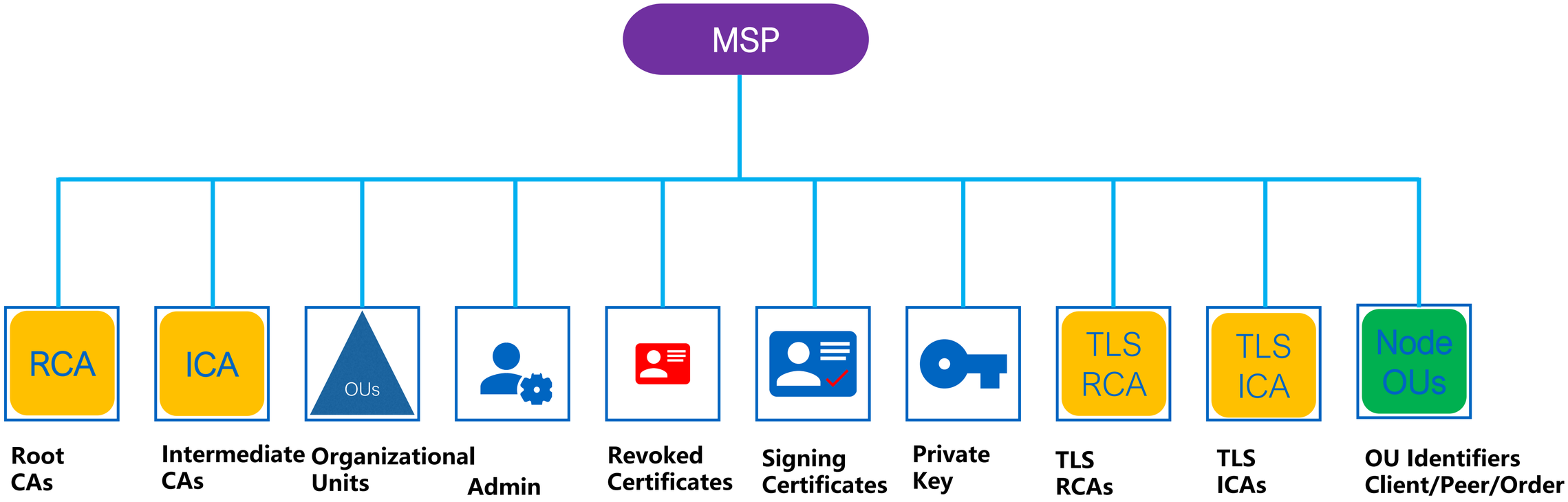




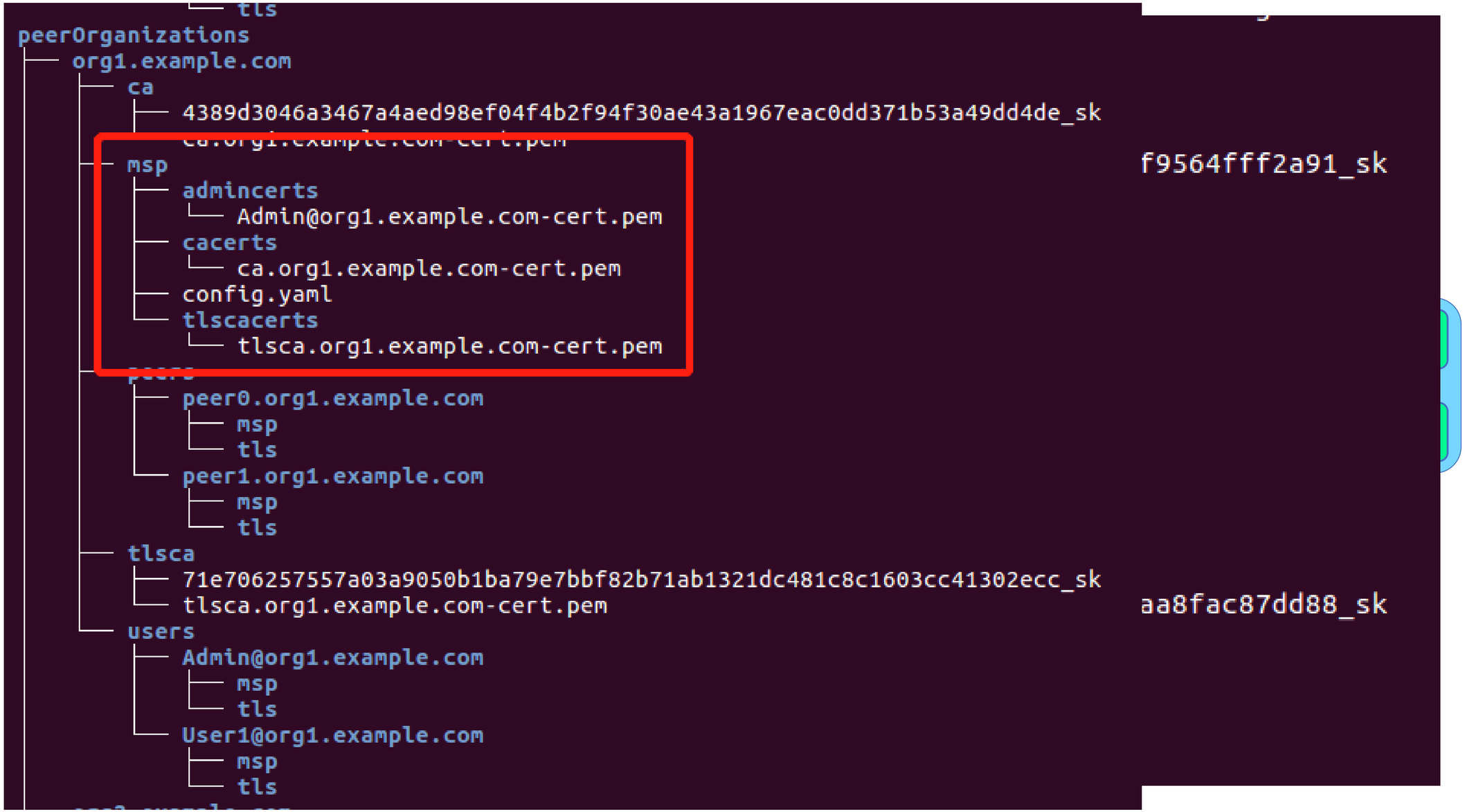
Revoked Certificate List



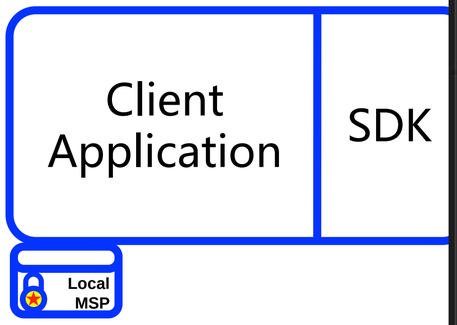
MSP Structure



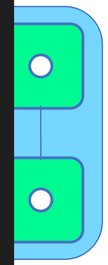
MSP的实现——Local MSP



Channel MS




```
Organizations:

    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg

        # ID to load the MSP definition as
        ID: OrdererMSP

        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/example.com/msp

    - &Org1
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: Org1MSP

        # ID to load the MSP definition as
        ID: Org1MSP

        MSPDir: crypto-config/peerOrganizations/org1.example.com/msp

        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: peer0.org1.example.com
              Port: 7051
```



Membership Service Provider

Multichannel Consensus

Private Data

Channel

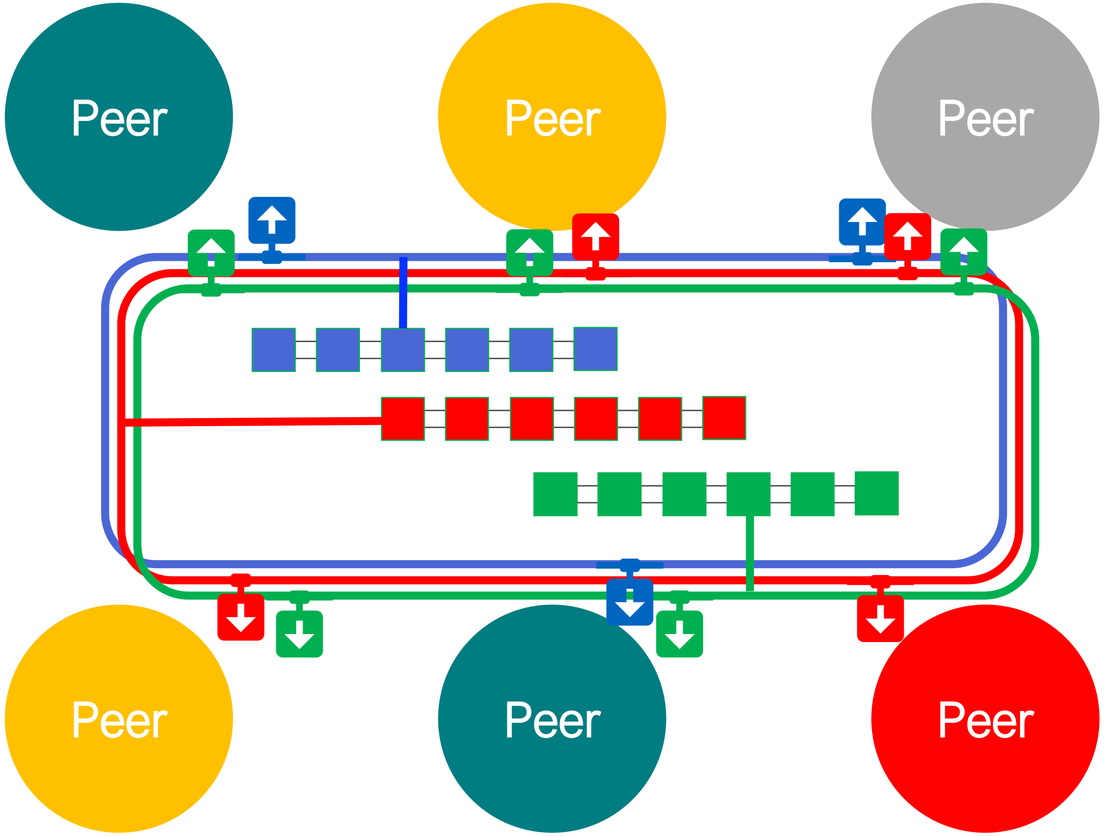
1. 特定的业务只和合作伙伴有关

2. 业务隔离，交易隐私

3. 每个Peer可以加入一个或者多个Channel

4. 每个channel对应一个账本，不同的账本数据是独立

5. Chaincode需要在channel上实例化

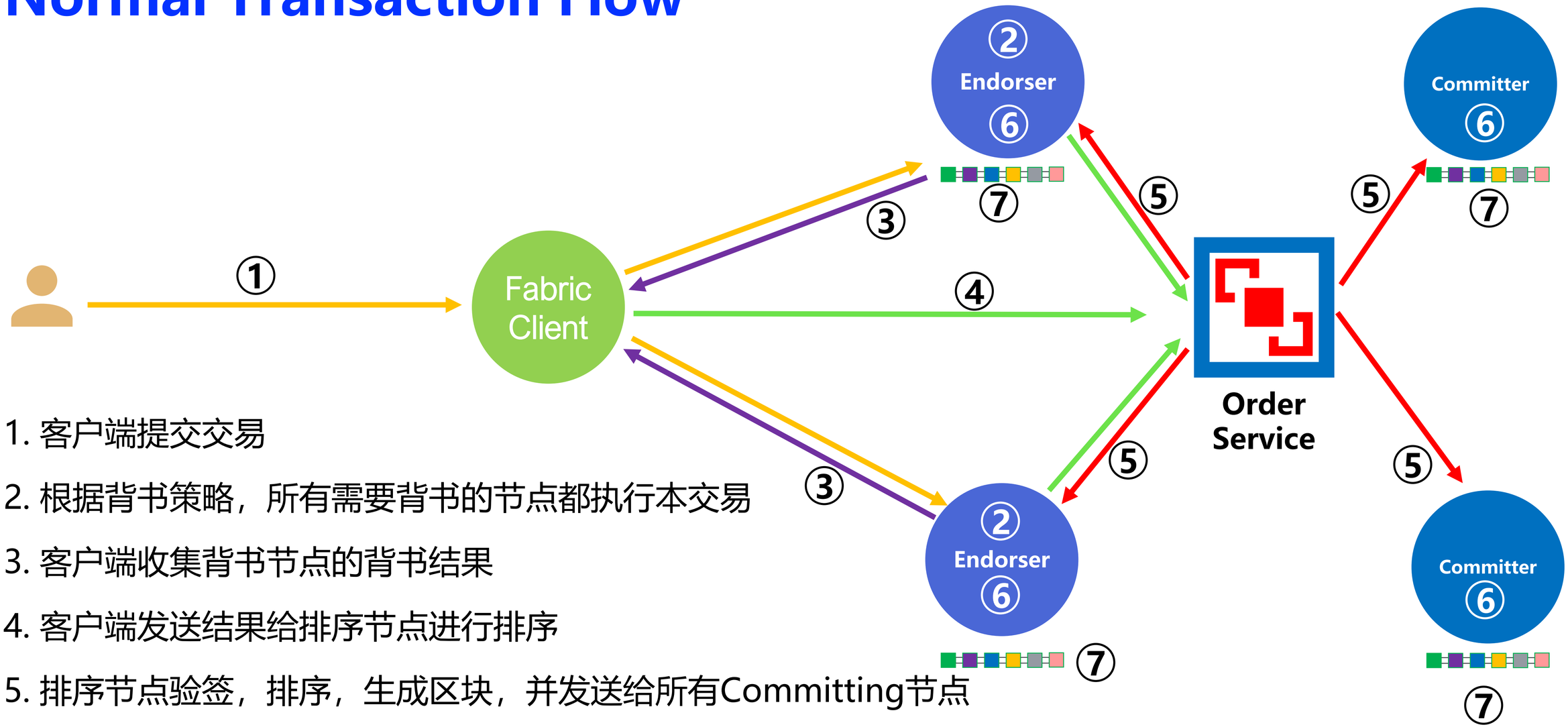


Membership Service Provider

Multichannel Consensus

Private Data

Normal Transaction Flow



1. 客户端提交交易
2. 根据背书策略，所有需要背书的节点都执行本交易
3. 客户端收集背书节点的背书结果
4. 客户端发送结果给排序节点进行排序
5. 排序节点验签，排序，生成区块，并发送给所有Committing节点
6. Committing Peer 验证交易
7. Committing Peer 把区块写入账本

Normal Consensus

Endorser

- 根据背书策略决定哪些Peer参与背书
- 客户端收集背书结果

Order

- 排序节点对所有交易进行排序已经生成block
- 排序节点把block发给提交节点验证

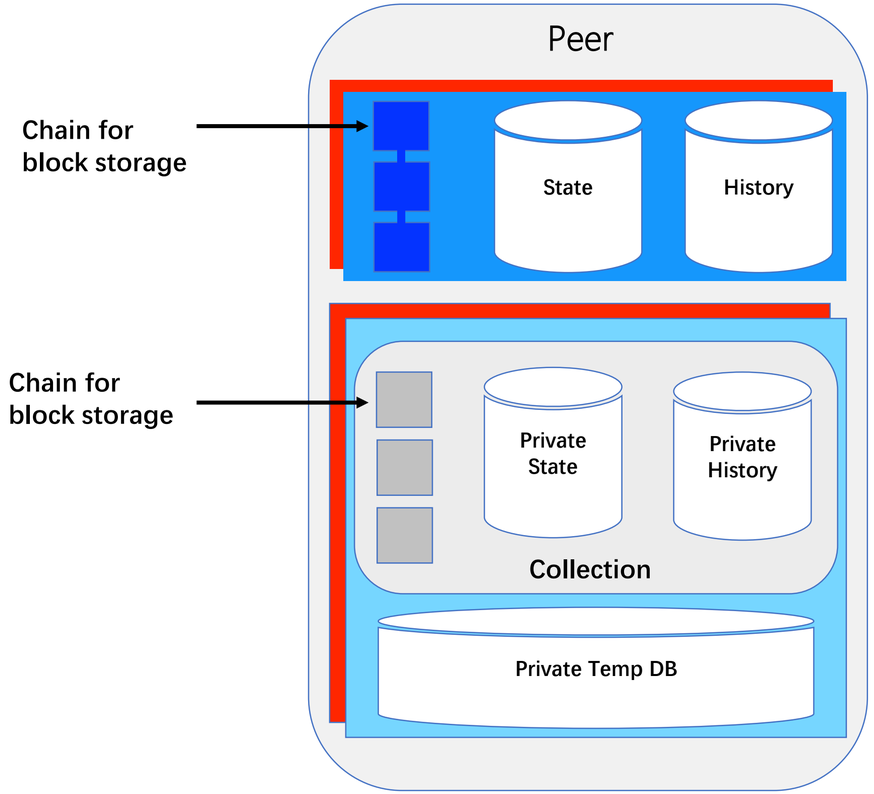
Validate

- 提交节点验证交易，背书策略，R/WSets
- 提交节点把block写进账本

Change on Ledger

1. Private Temp DB - stores transient (uncommitted) private read write sets for transactions ‘on the side’, between endorsement time and commit time. Keyed by txid.

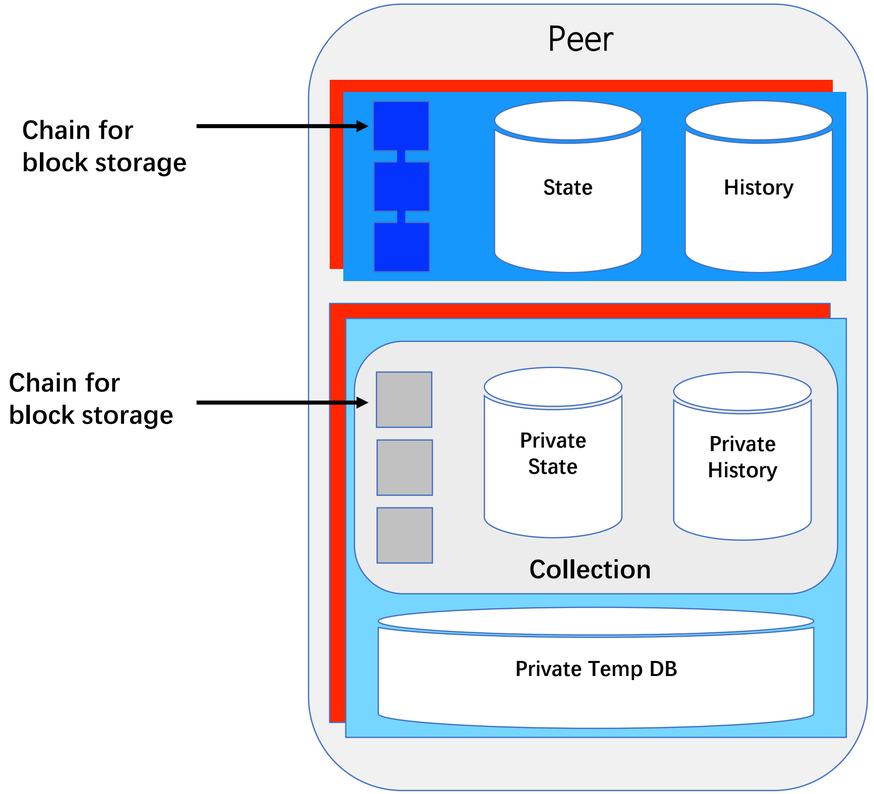
2. Private write-set log - Primary storage for committed private write sets, a transaction log of private write sets keyed by blockNum or (blockNum:tranNum) to assist in state transfer alongside blocks (which is the transaction log for public data).



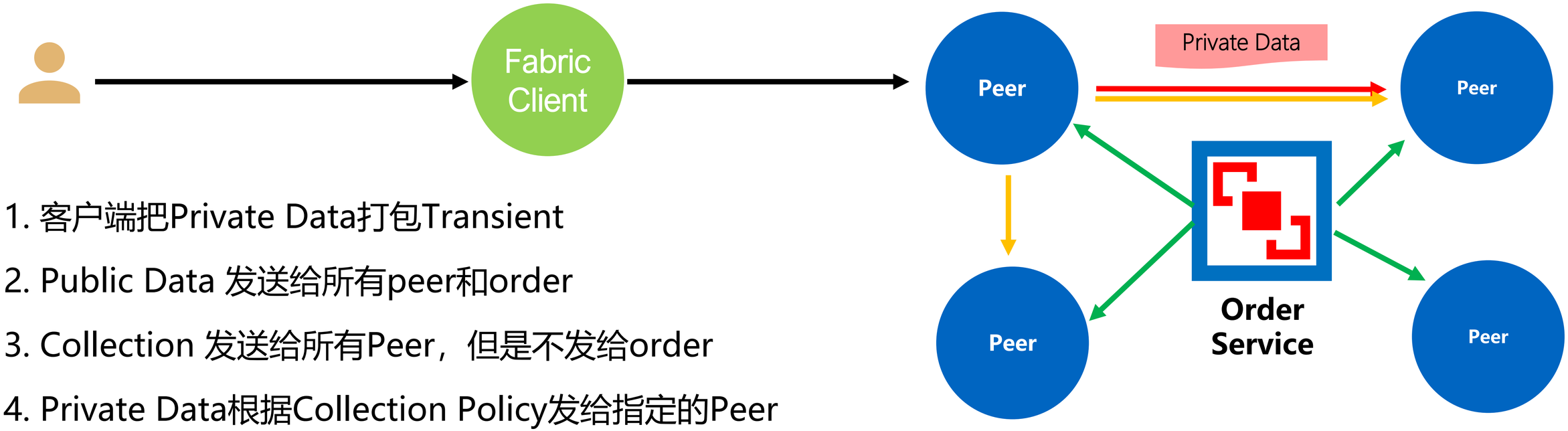
Change on Ledger

3. Private State DB - Similar to state DB - stores latest version of committed private keys/values. Used by chaincode APIs. Can be rebuilt from private write-set log, just like the normal state db can be rebuilt from normal block storage. Keyed by chaincodeid:key, same as normal state DB.

4. Private History DB - stores history of committed private value updates of a key (pointer to private write-set log blockNum:tranNum). Keyed by chaincodeid:key:blockNum:tranNum, same as normal history DB. Used by GetHistoryForKey chaincode API.



Change on Transaction



1. 客户端把Private Data打包Transient
2. Public Data 发送给所有peer和order
3. Collection 发送给所有Peer，但是不发给order
4. Private Data根据Collection Policy发给指定的Peer

Change on Consensus

Endorsement Phrase:

1. Primary R/W Set, public data as normal transaction
2. Hashed R/W Set, hash of both key and value, stored in block
3. Private data stored on Privat Temp DB

Validation Phrase:

1. Validate primary read set and hashed read set against State DB
2. Validate hash of private read set against private temp DB if exist
3. Validate hash of private read set against local private temp DB if exist
4. Otherwise, gossip from other peer