

Hyperledger Fabric:

Crittografia per blockchain *permissioned*

Giovanni Schmid
Istituto di Calcolo e Reti ad Alte Prestazioni
Consiglio Nazionale delle Ricerche

Introduction

Architecture and
Functions

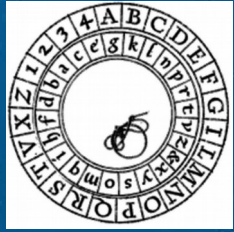
Permissioned
Networks

Code and data
security



Attività in essere all'ICAR

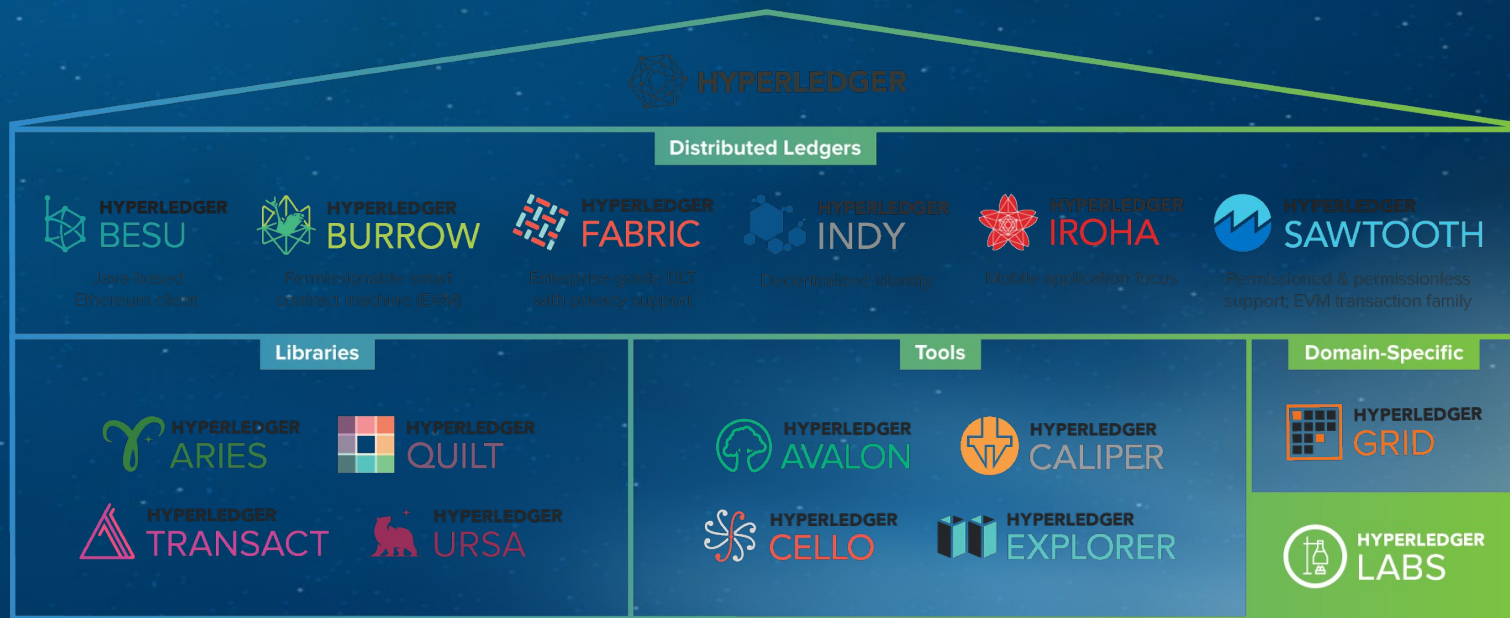
- **Progetto Napoli Blockchain** : Gruppo di studio volontario su blockchain (trasparenza) e criptovalute (pagamenti), Delibera di G.C. n. 465 del 05/10/2018
- **Blockchain ed anti-contraffazione** : Progettualità in corso di definizione con la Struttura *Valorizzazione della Ricerca* del Cnr e la società *Plug-in*
- **Attività di ricerca e sviluppo nel contesto della Medicina Digitalizzata** : Gestione ed innovazione di processi sanitari con l'ausilio di un sistema blockchain (progetto FSE7)



Hyperledger Fabric Architecture and Functions



Fabric and Hyperledger





Underlying philosophy

There will not be only one blockchain, or a chain-of-all-chains.

There will be many public chains and millions of private chains, potentially each with a different consensus mechanism, preferred smart contract language / mechanism, and other characteristics.

The more common code underlying these chains, the better for everyone.

This is still early days – perhaps like 1994 and the Web?

Arnaud Le Hors, March 2017

Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

Membership

Chaincode lifecycle

Private data



Fabric key features

- Support for multiple credential and cryptographic services for identity
- Reflect business processes by specifying who endorses transactions
- Do not require any built-in cryptocurrency
- Eliminate non deterministic transactions
- Scale the number of users and transaction throughput
- Implement the very general concept of asset
- Support rich data queries of the ledger
- Dynamically upgrade fabric and chaincode
- Pluggable consensus protocol
- Support for data and process confidentiality

Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

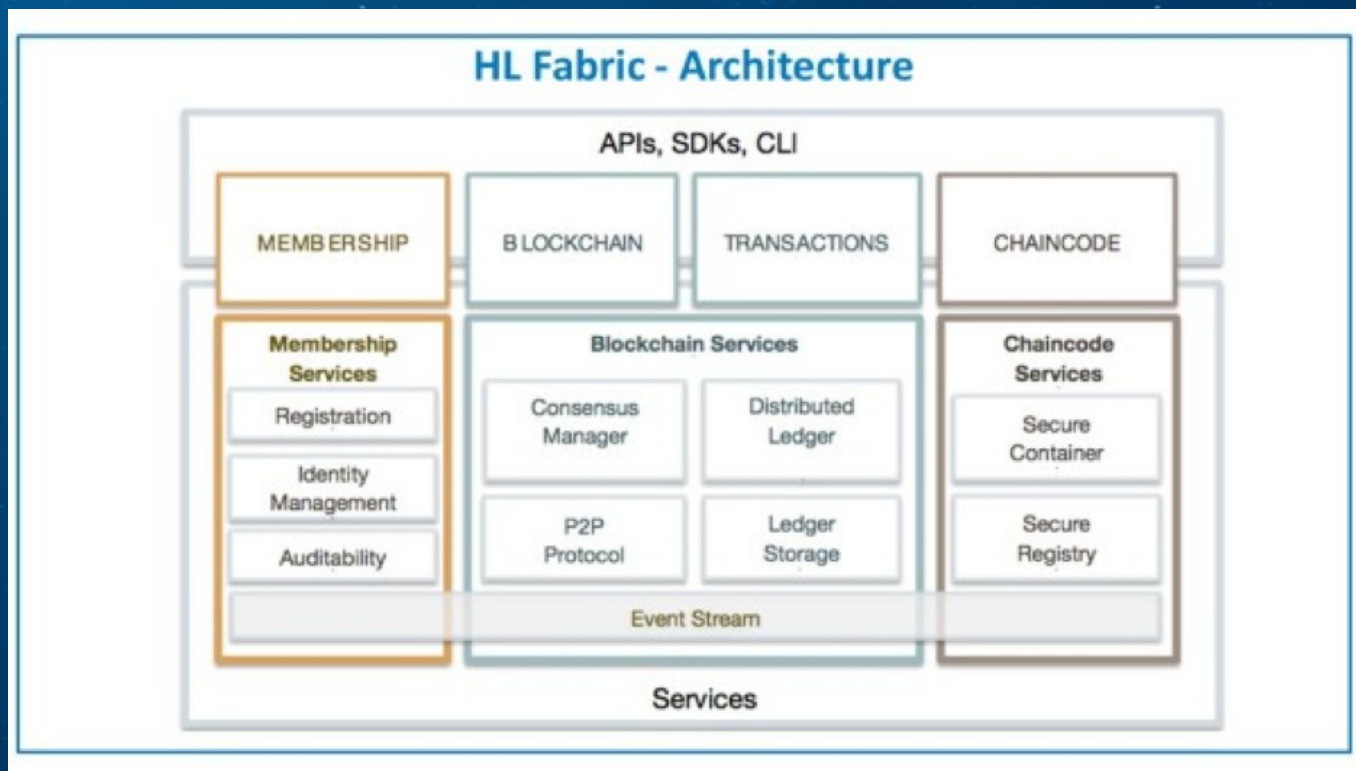
Membership

Chaincode lifecycle

Private data

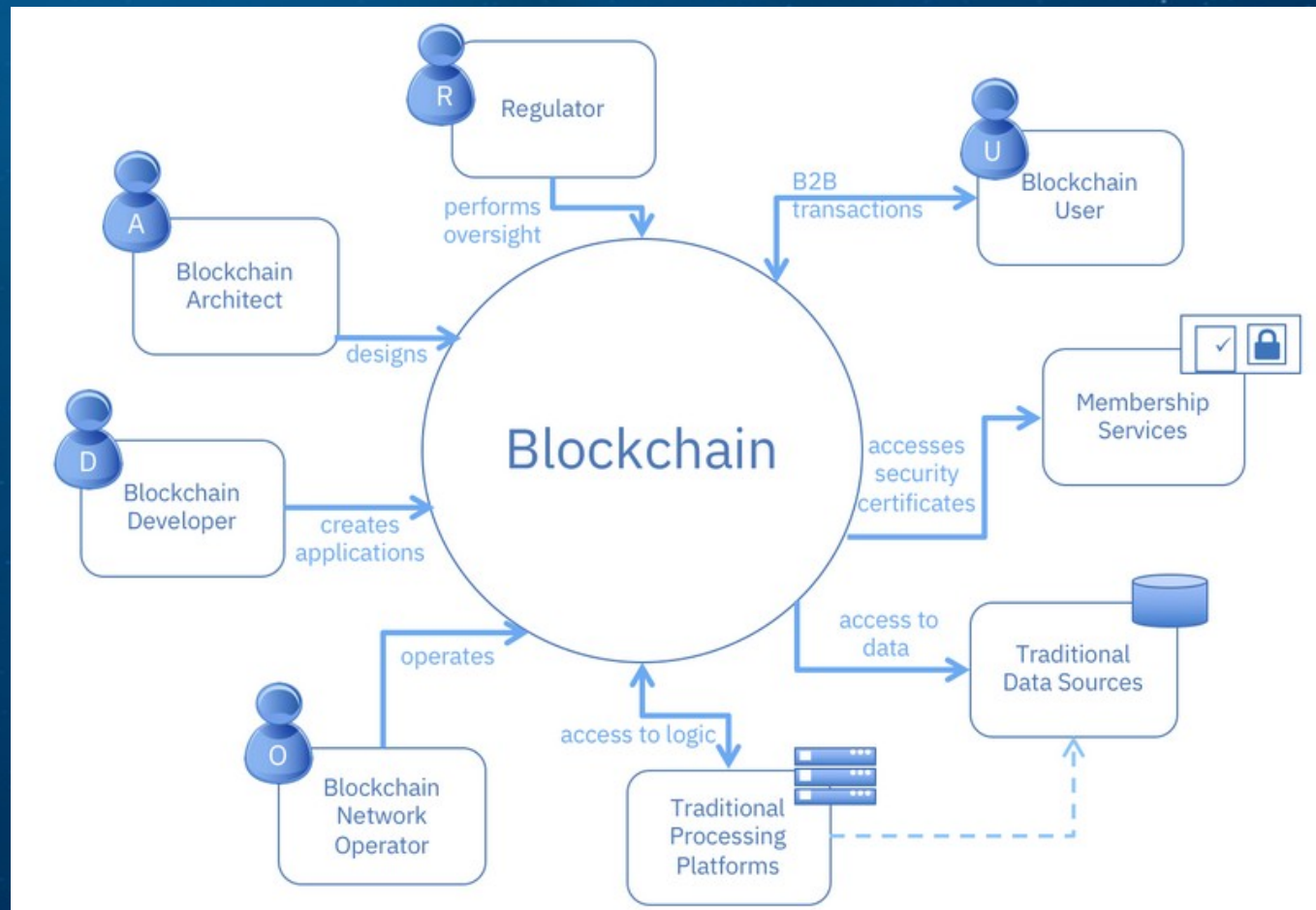


Fabric architecture



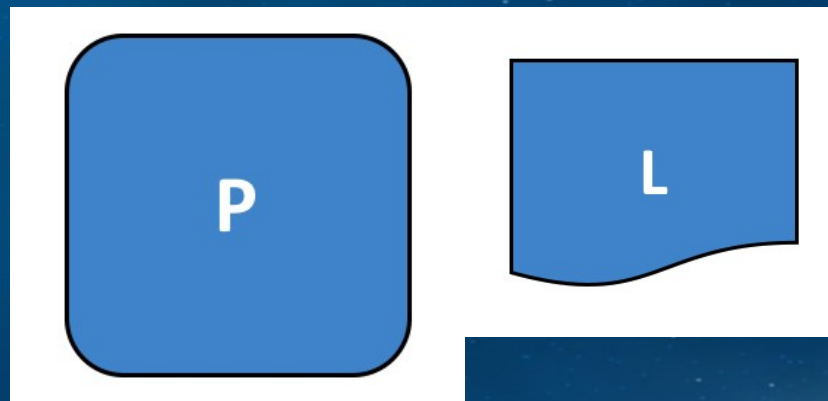


Actors in a Fabric network





Peers and their roles



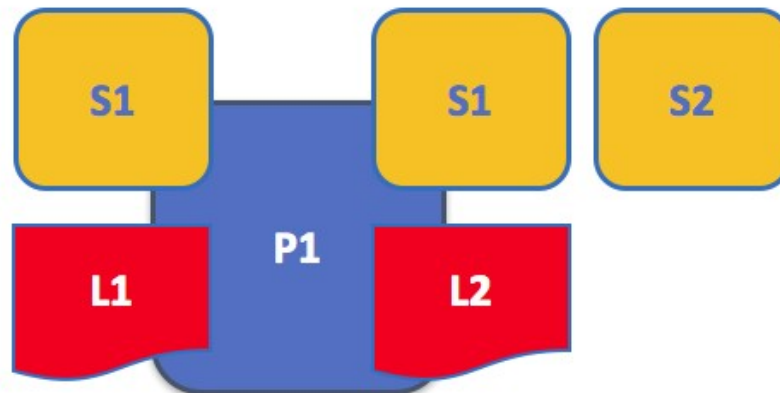
Anchor peers are used by the gossip protocol to make sure peers in different organizations know about each other.

Leading peers communicate with the network ordering service on behalf of the organization.

Endorsers execute a chaincode transaction and return a proposal response to the client application.



Peers, Chaincode and Ledger

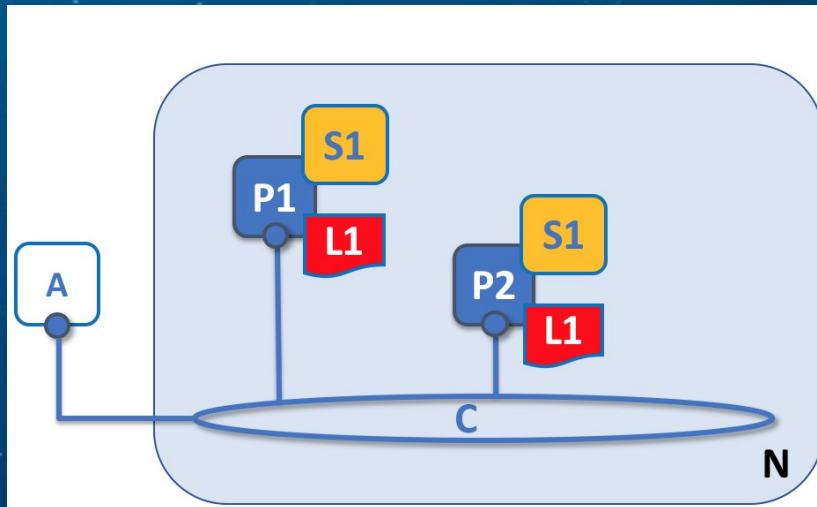









Peers host one or more ledgers, and each ledger has zero or more chaincodes that apply to them.

When a peer is first created, it has neither ledgers nor chaincodes.



Peers and Channels



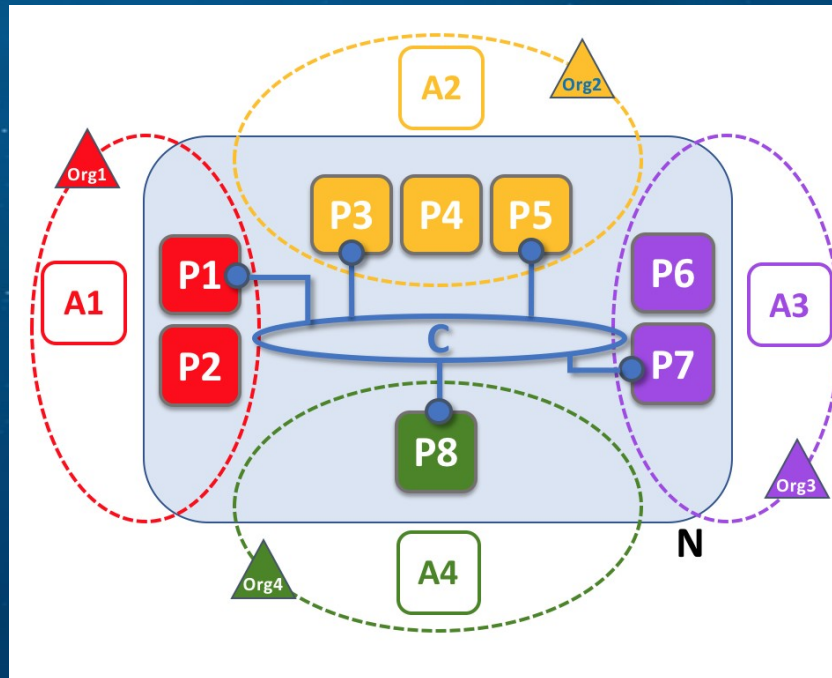
	Blockchain Network		Ledger
	Channel		Application
	Peer		Principal PA (e.g. A, P1) communicates via channel C.
	Chaincode		








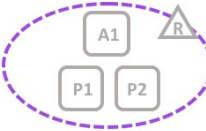
Peers interact with each other, and with applications, via channels.

A **channel** is defined by a configuration block, it composes of a channel-specific ledger and a channel-specific ordering service. Transacting parties must be properly authenticated to a channel in order to interact with it.



Peers and Organizations



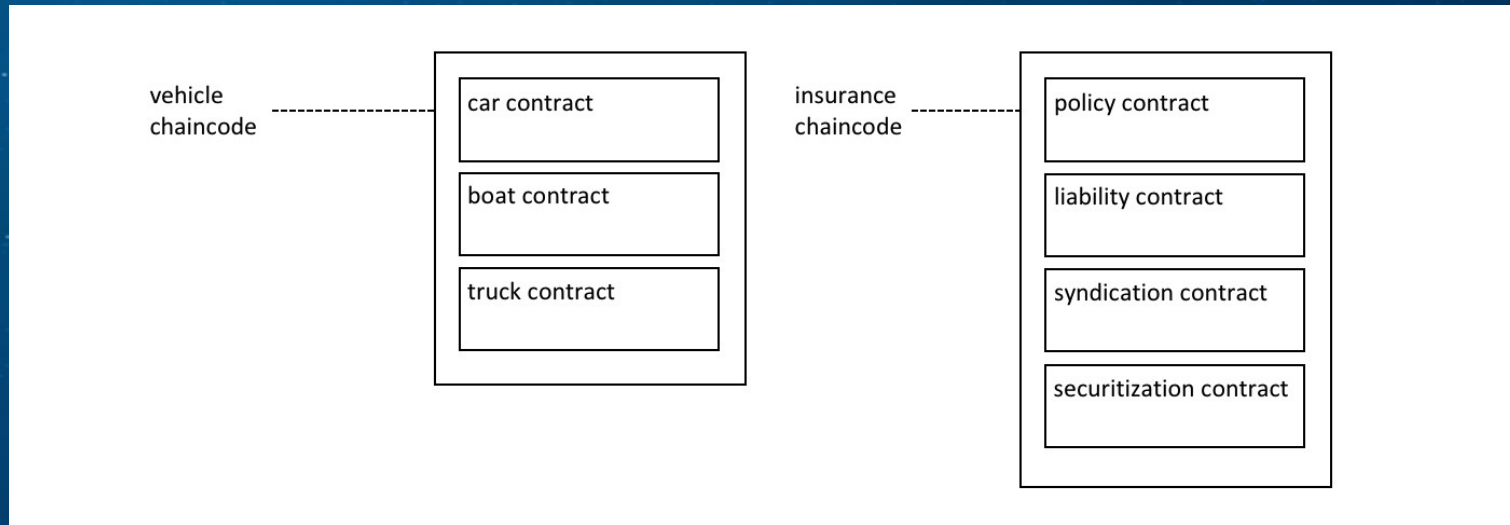
	Blockchain Network		Ledger
	Channel		Application
	Peer		Principal PA (e.g. A1, P5) communicates via channel C.
			Organization
		Organization R owns application A1 and peers P1, P2.	

The blockchain network is built up from the peers owned and contributed by the different organizations.

Peers have an identity assigned to them via a digital certificate from a certificate authority.



Smart contract and Chaincode



Application chaincode:

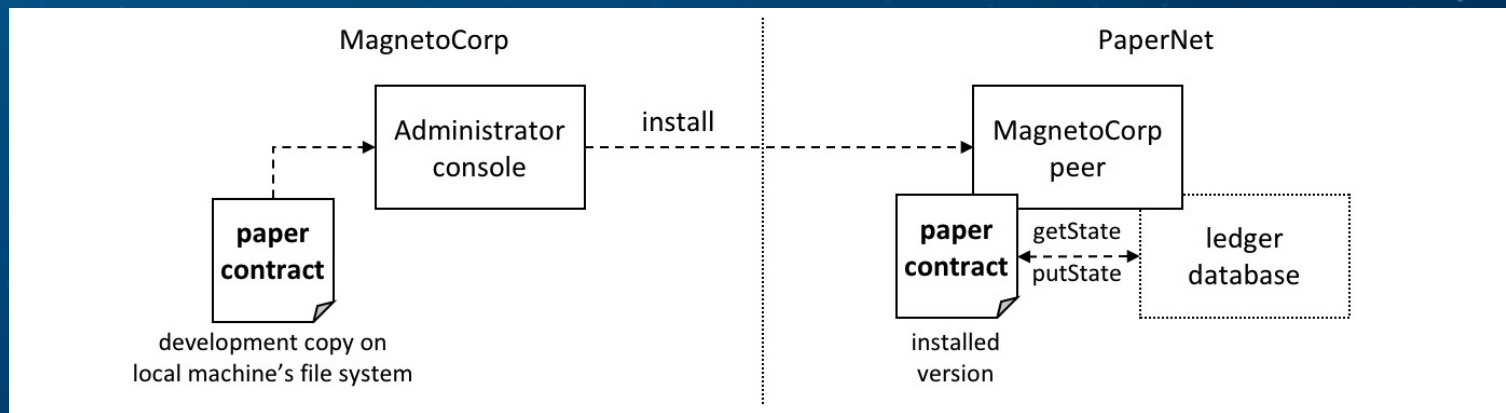
a program (Go, Node.js, Java) implementing a prescribed interface.

runs in a secured Docker container isolated from the endorsing peer process.

initializes and manages ledger state through transactions submitted by applications.

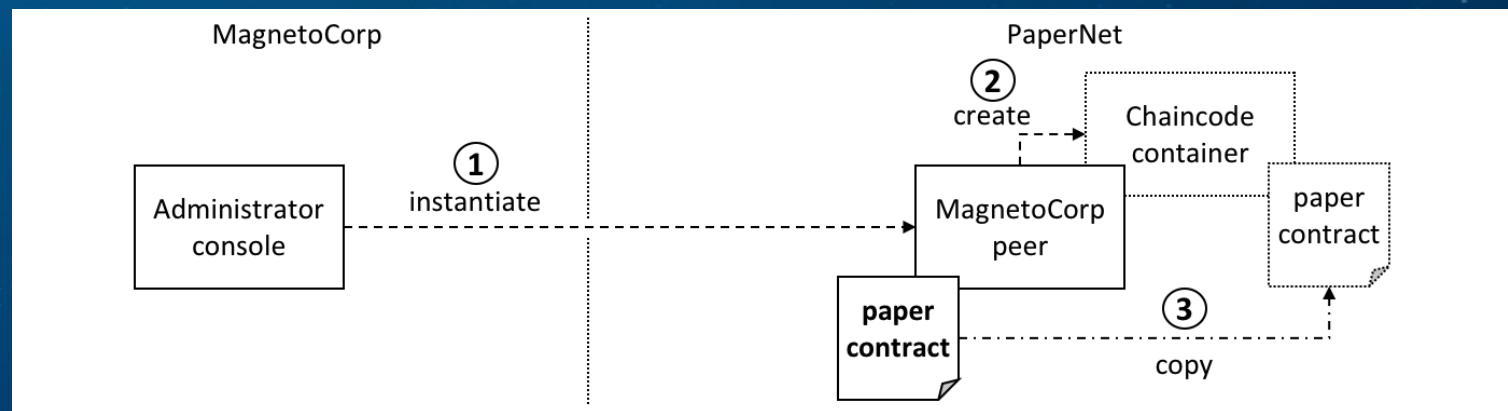


Install Chaincode





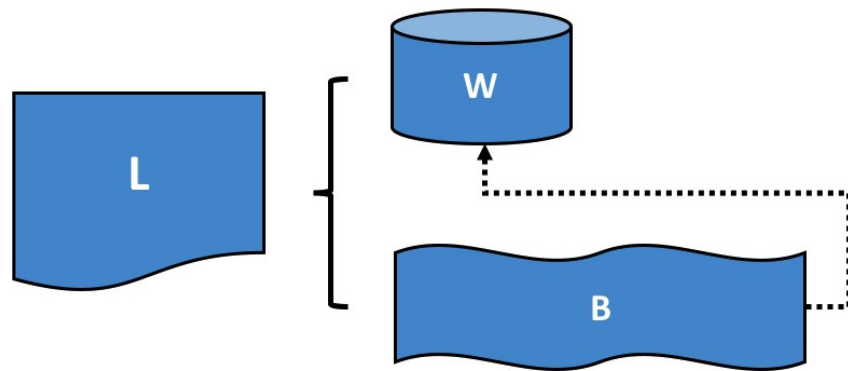
Instantiate Chaincode








Endorsement policy: defines the peer nodes on a channel that must execute transactions attached to a specific chaincode application, and the required combination of responses (endorsements).



Fabric Ledger



	Ledger
	World State
	Blockchain
	L comprises B and W
	B determines W

Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

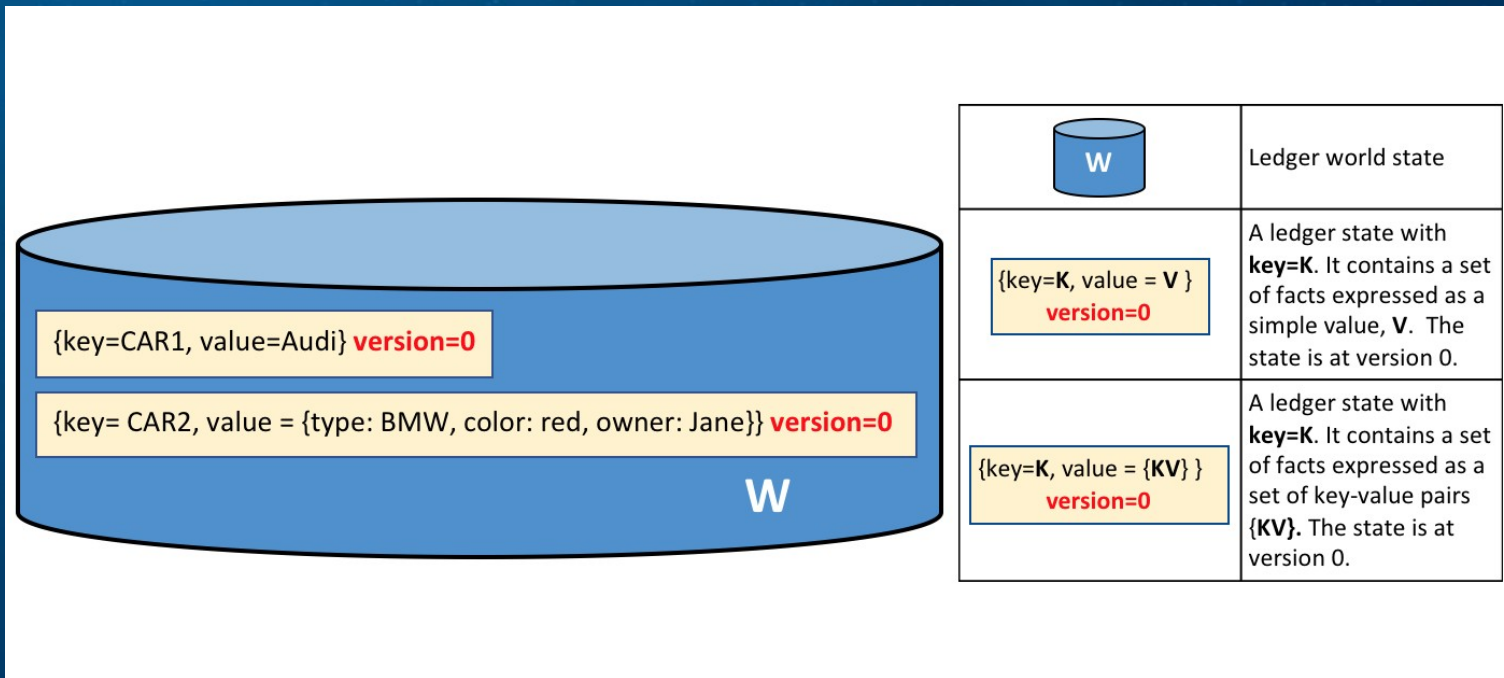
Membership

Chaincode lifecycle

Private data



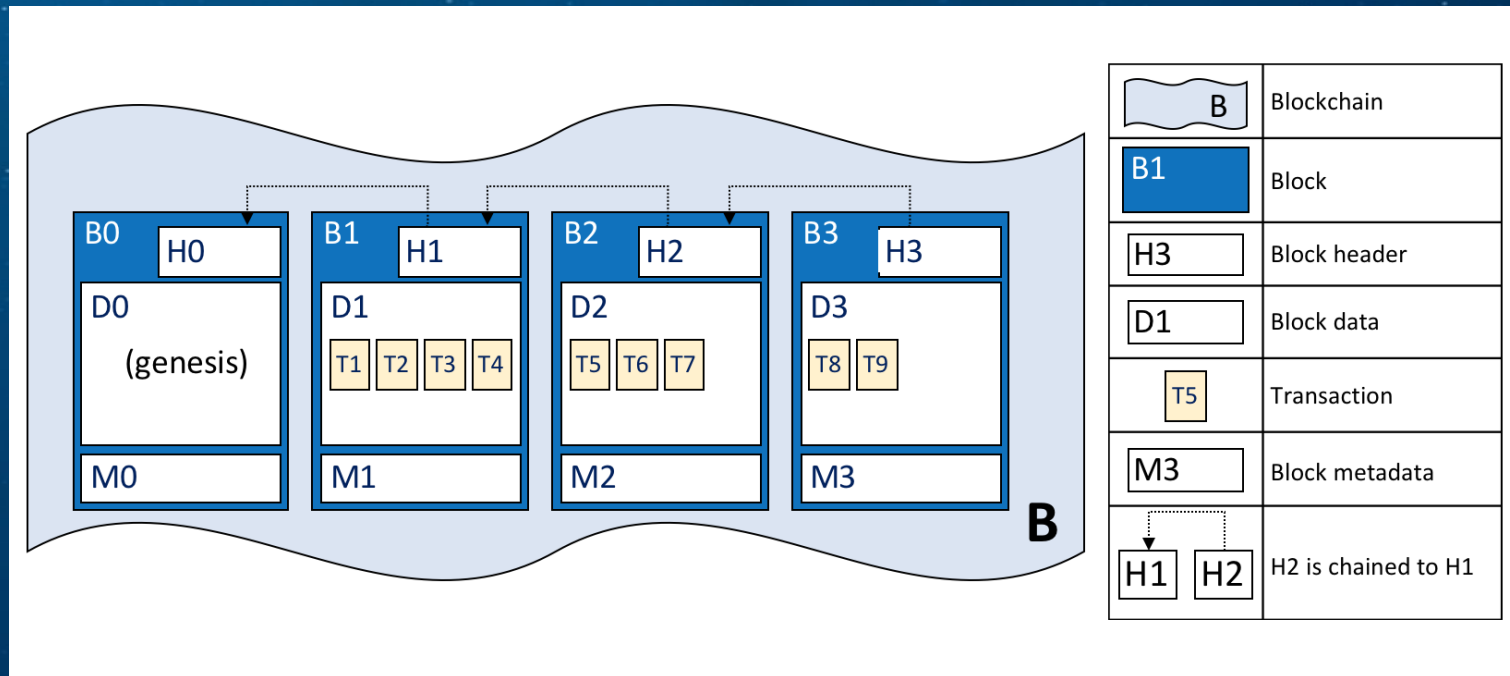
World State database



The **world state** represents the latest values for all keys included in the chain transaction log, where keys represent **assets** and values encode their **states**.



Blockchain



Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

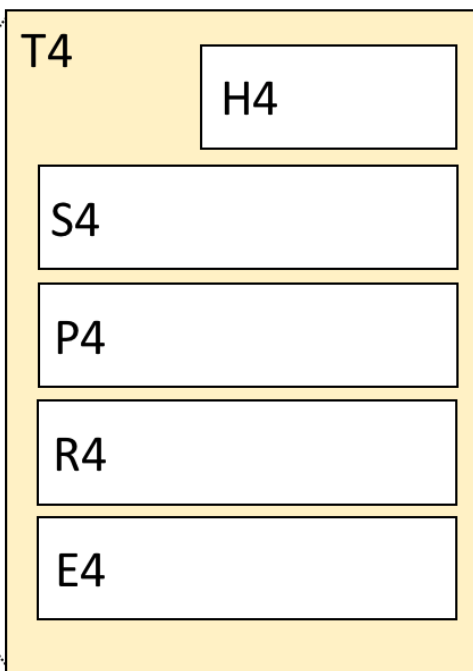
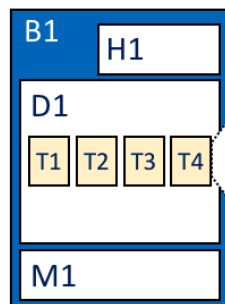
Membership


Chaincode lifecycle

Private data



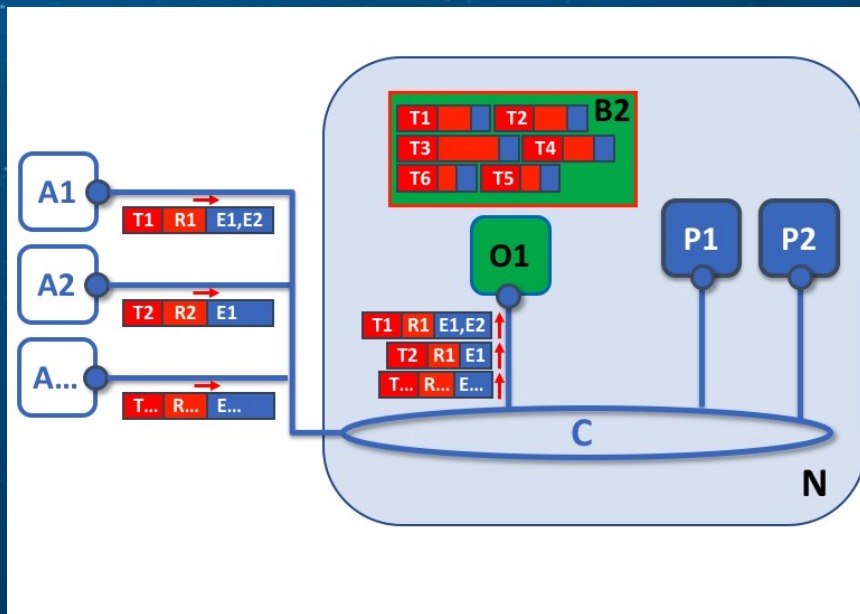
Transactions








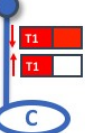



T4	Transaction
H4	Header
S4	Signature
P4	Proposal
R4	Response
E4	Endorsements
T4 	V4 V4 is detailed view of T4



Orderers



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



Orderers and Consensus



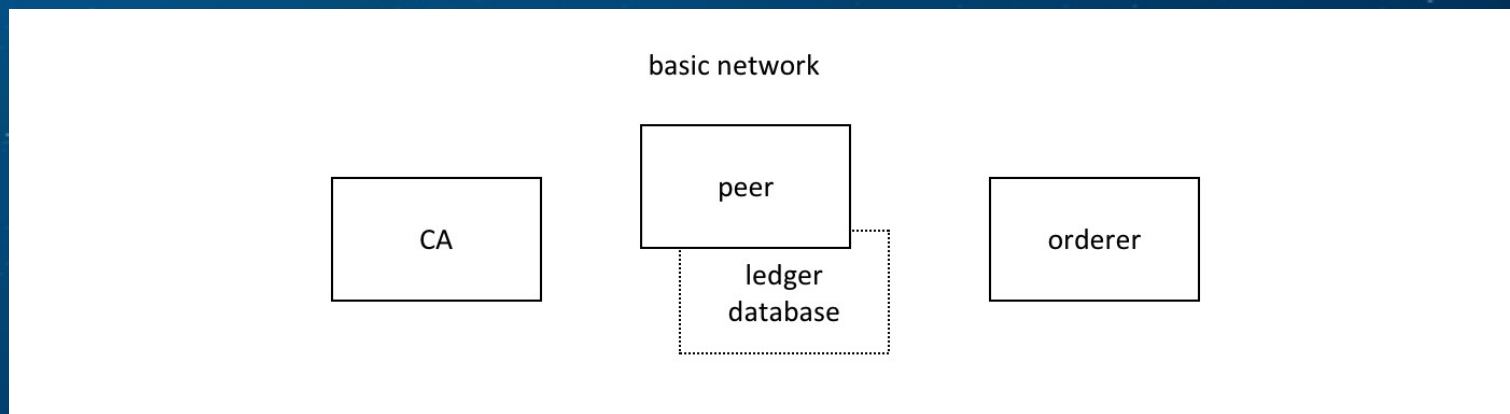
- 1982: Lamport, Shostak, Pease (BGP)
- 1990: Lamport (algoritmo Paxos)
- 1999: Castro, Liskov (algoritmo PBFT)
- 2008: Nakamoto (PoW + consenso)
- 2014: Bessani, Sousa, Alchieri (BFT-SmaRt)
- 2017: Micali, Vaikuntanathan (Algorand BFT)
- 2019: *Calibra* (Libra BFT)

Kafka: CFT protocol that uses a “leader and follower” node configuration.
Available since Fabric v1.0, but quite difficult to setup and administer

Raft : New as of v1.4.1, it is again a CFT “leader and follower” protocol.
However, it is easier to configure and manage than Kafka.



Basic Fabric network



Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

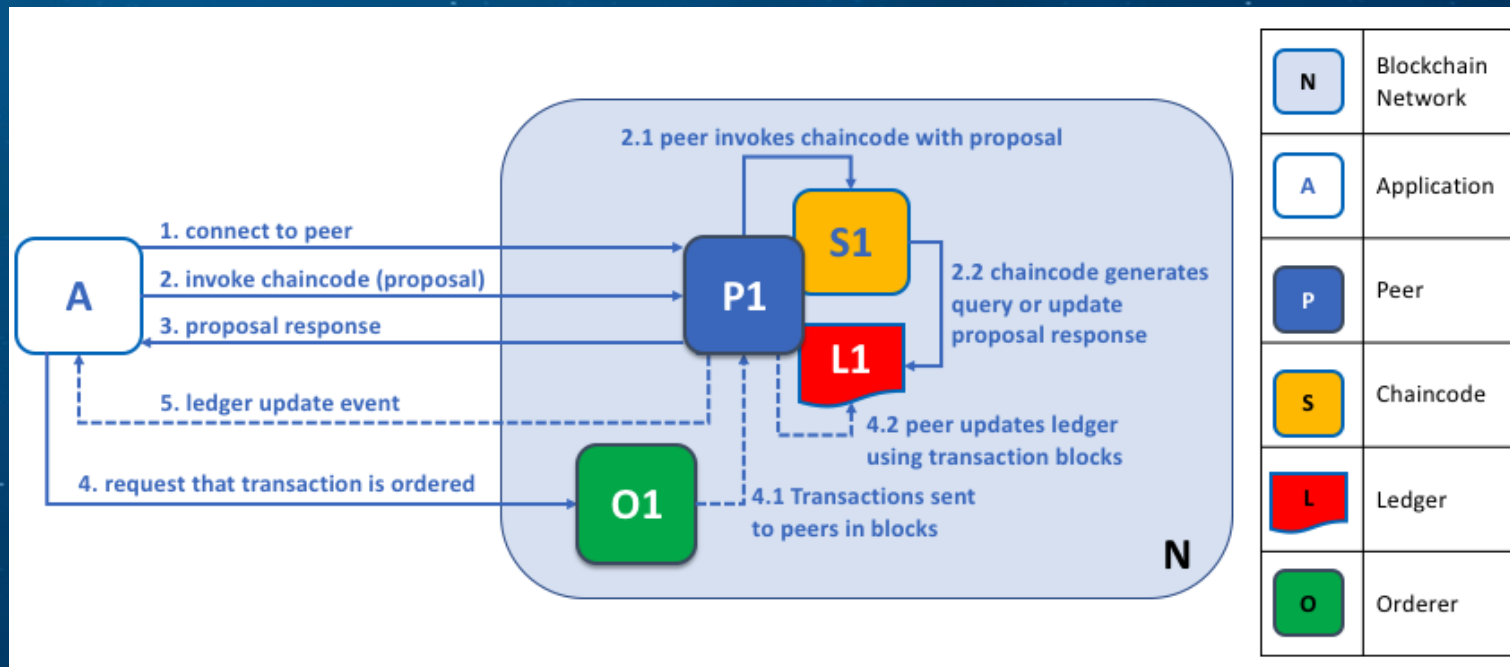
Membership

Chaincode lifecycle

Private data



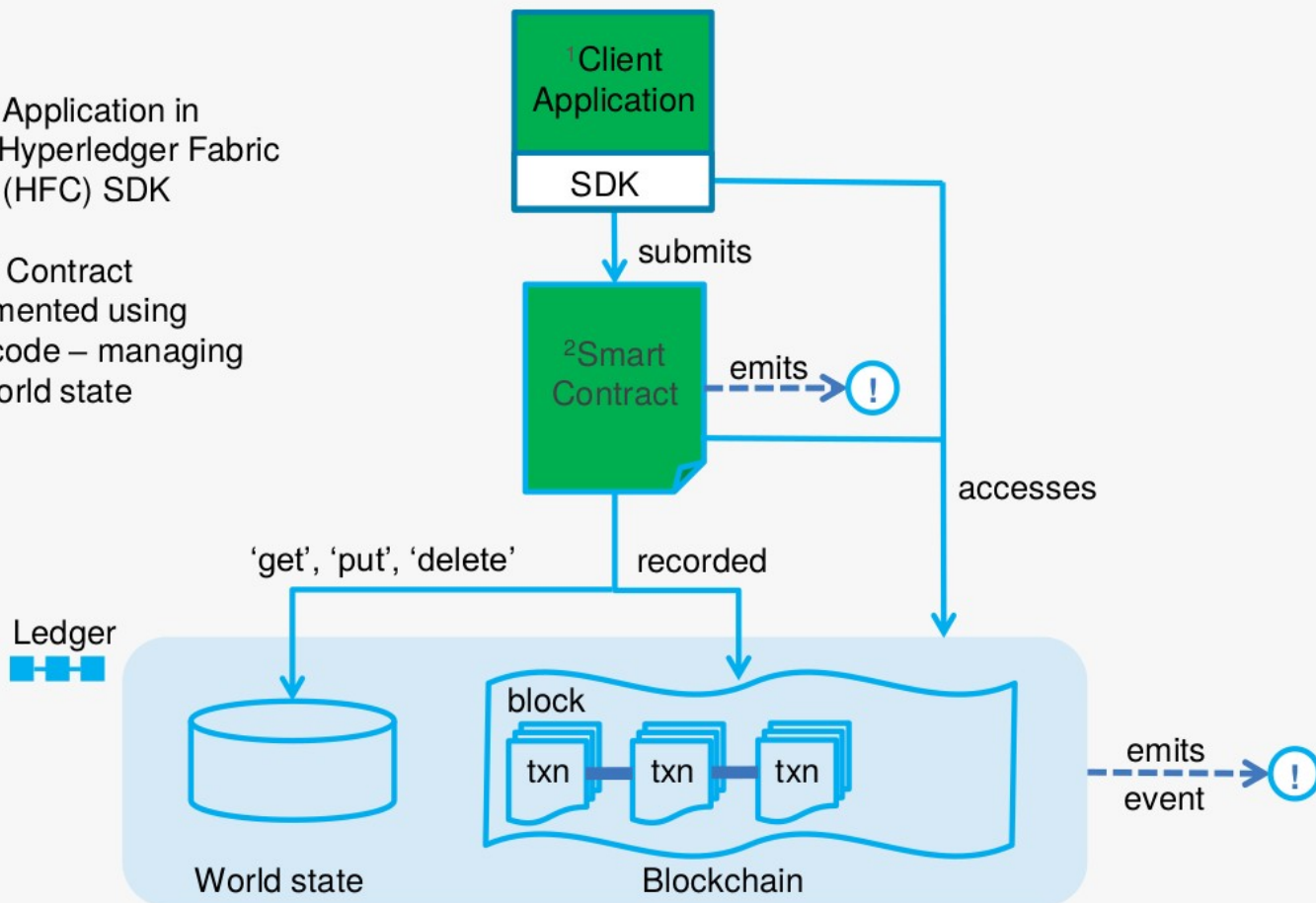
Application workflow

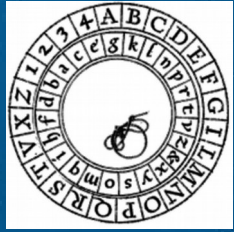




Application deployment

1. Client Application in using Hyperledger Fabric Client (HFC) SDK
2. Smart Contract implemented using chaincode – managing the World state

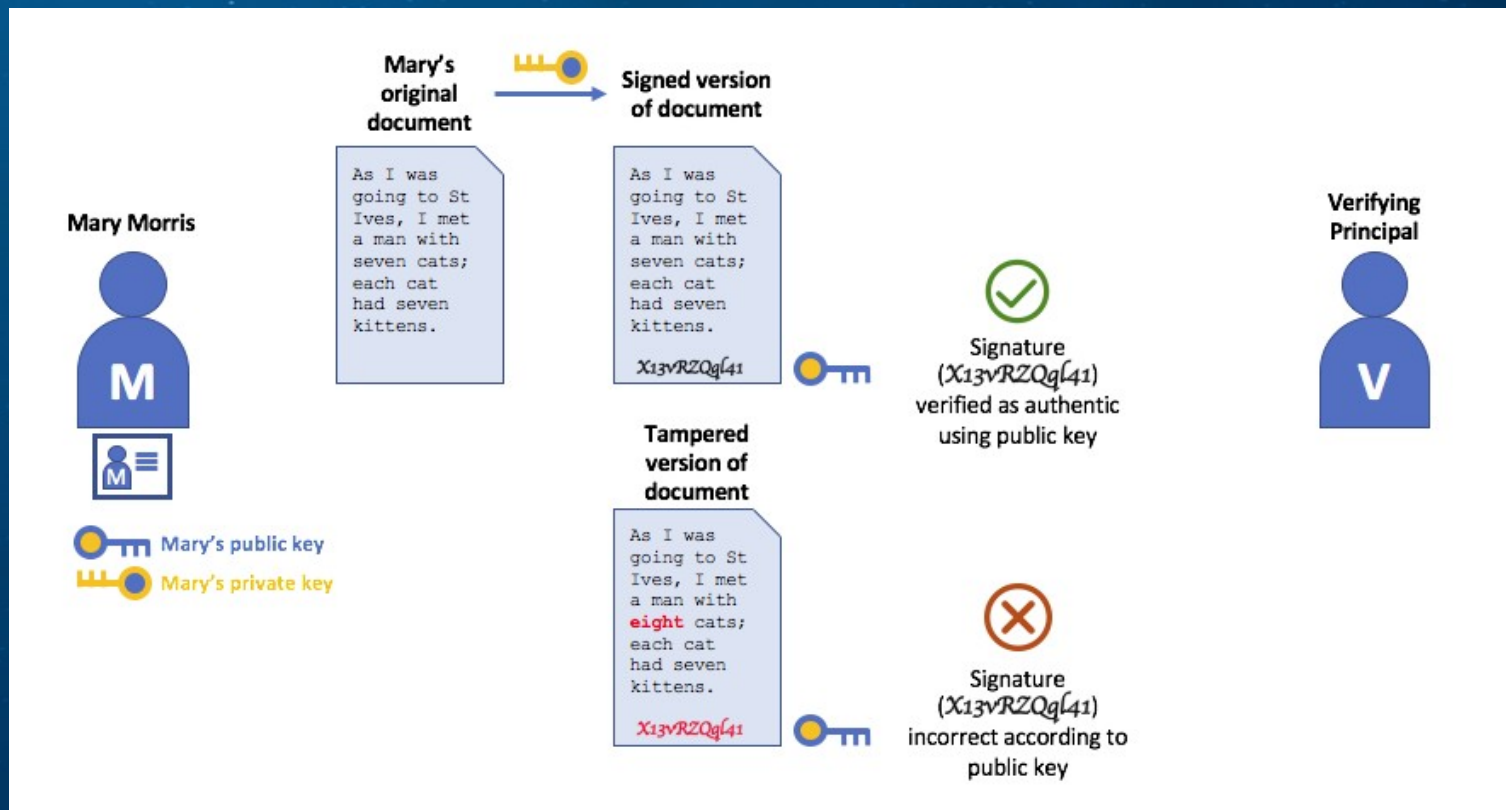




Permissioned Networks



Document authentication





Public Key Certificates

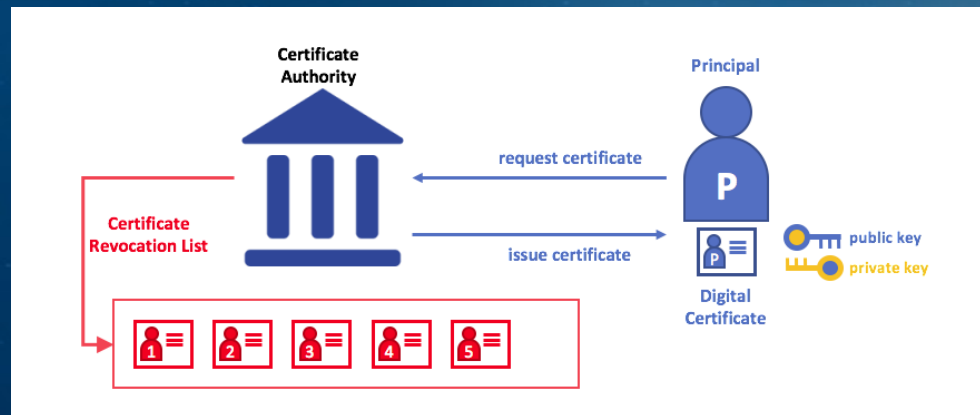
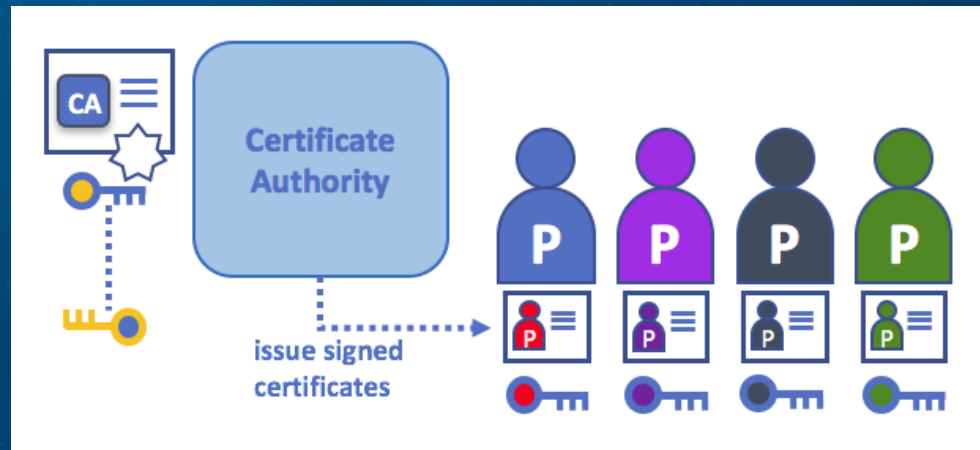
Mary Morris



```
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=Mitchell 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:
```

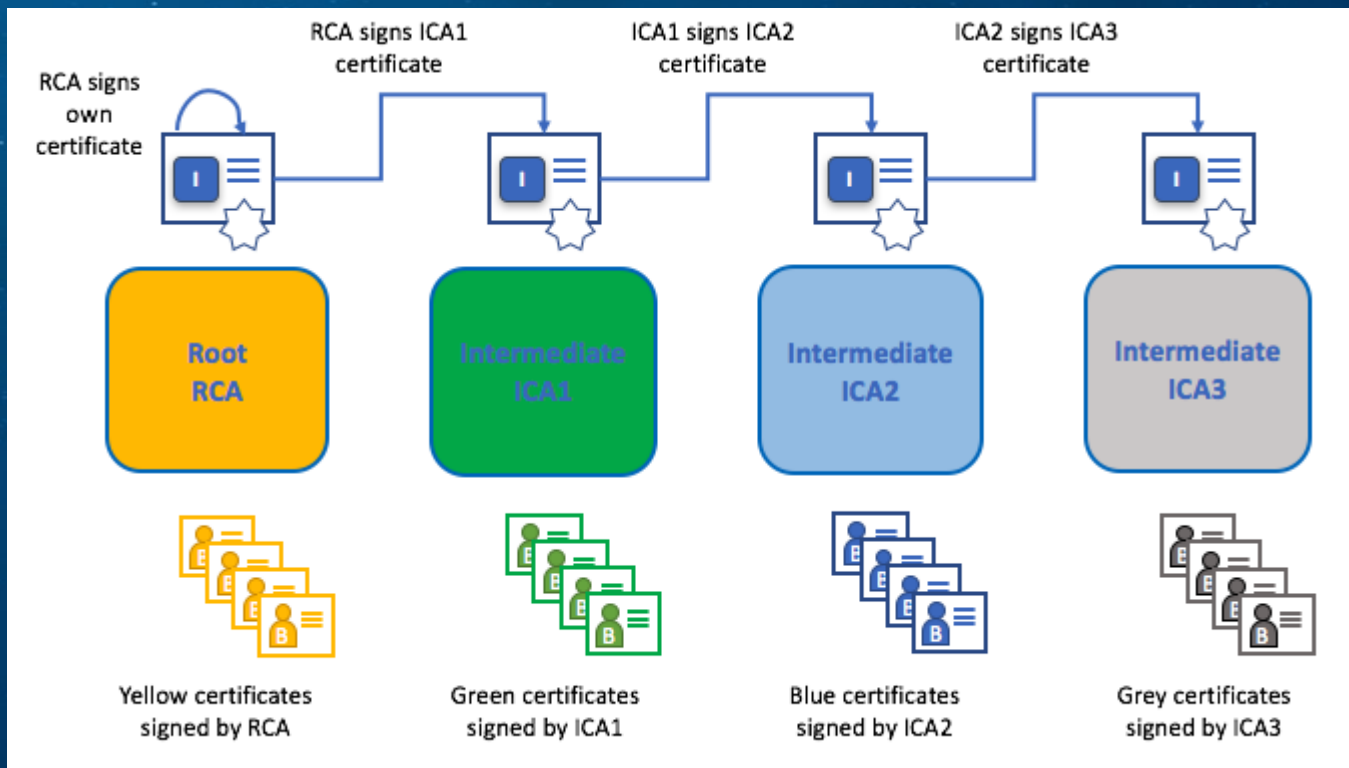


Certificate Authority



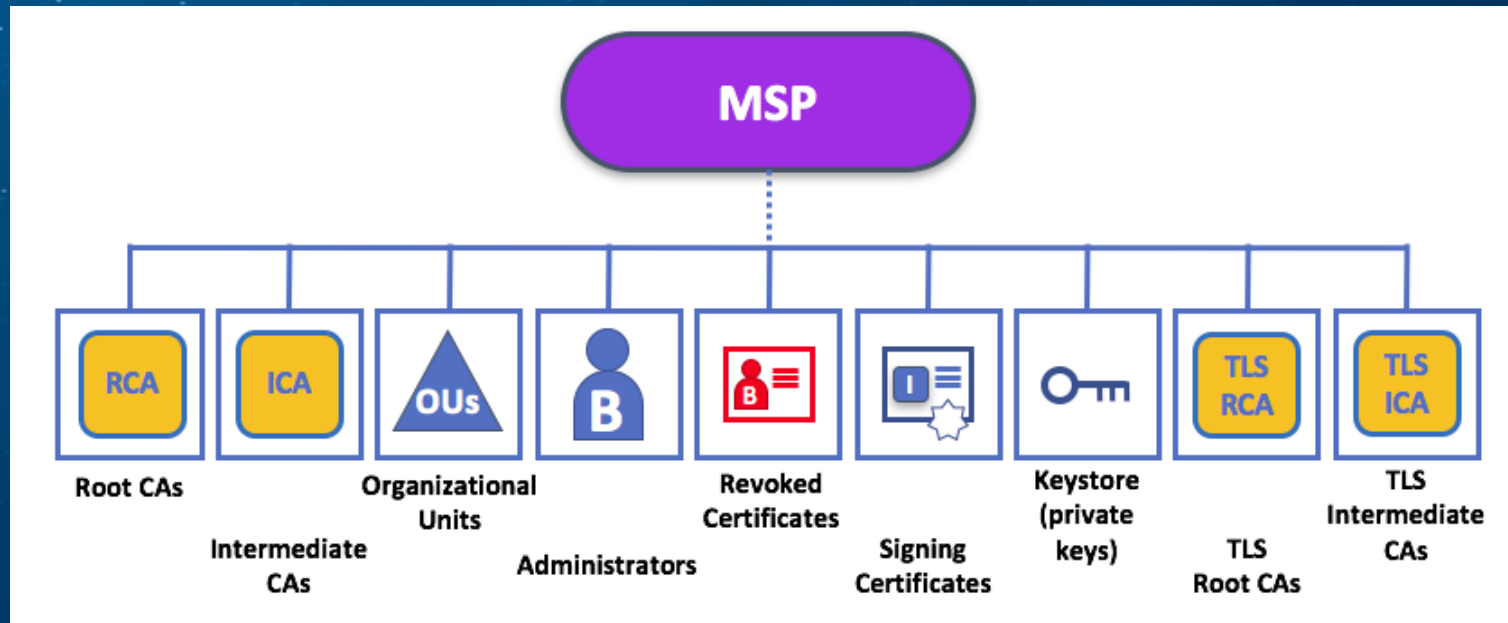


Chain of Trust





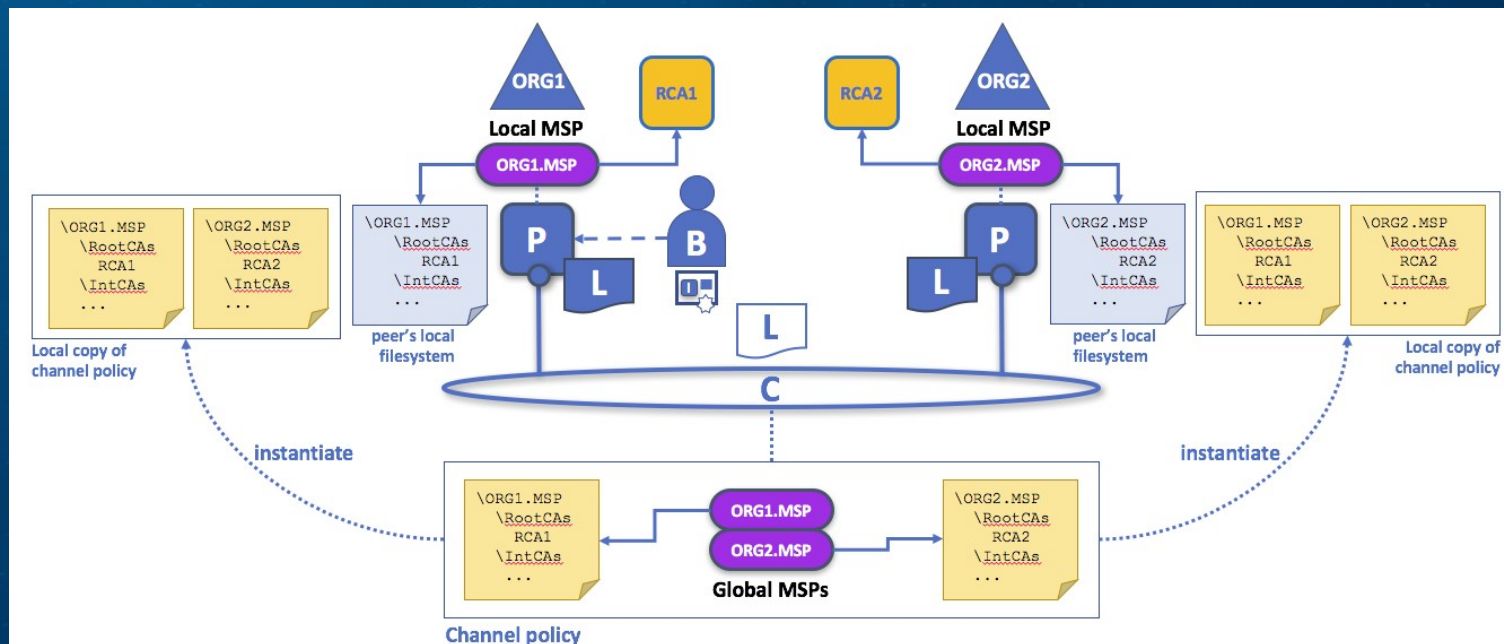
MSP structure



Membership Service Provider (MSP) identifies which Root CAs and Intermediate CAs are trusted to define the members of an organization

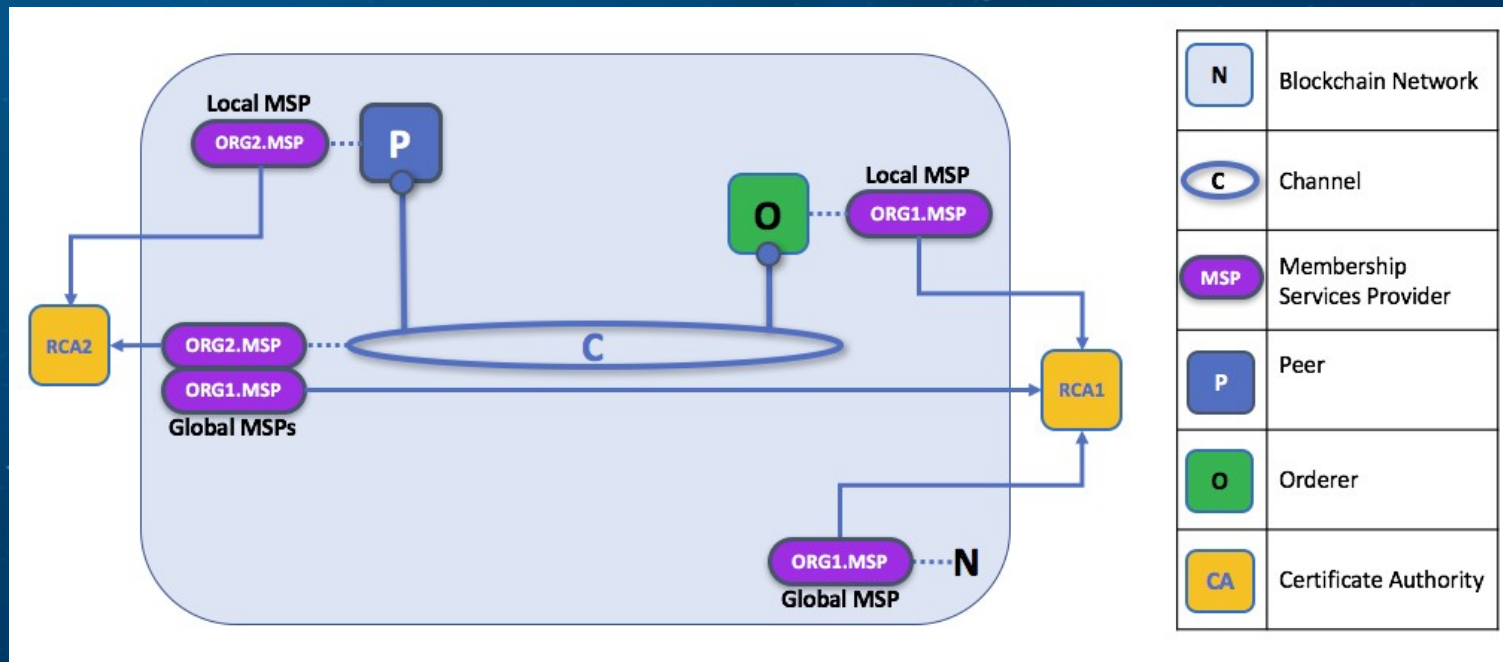


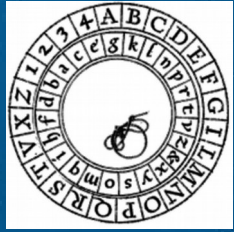
Local vs global MSP





MSP levels

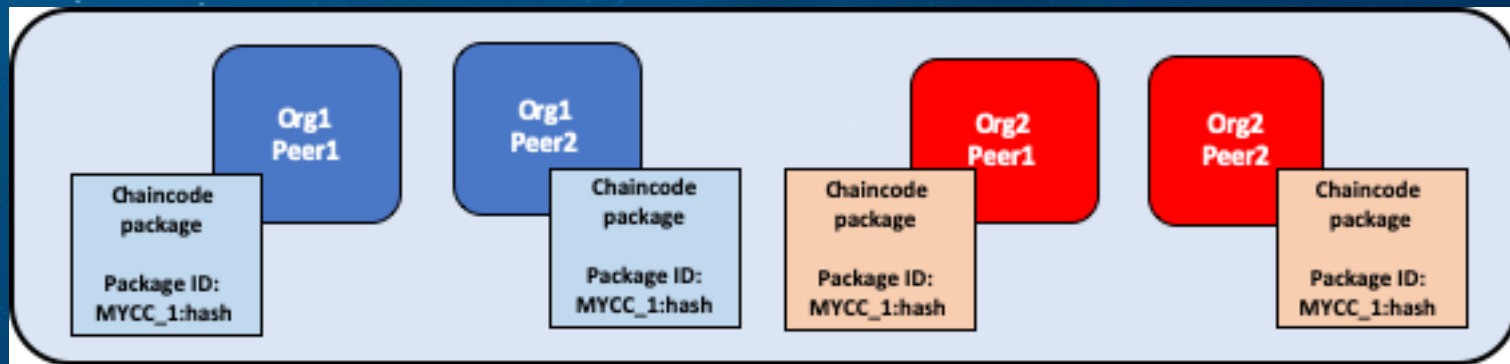




Code and data security

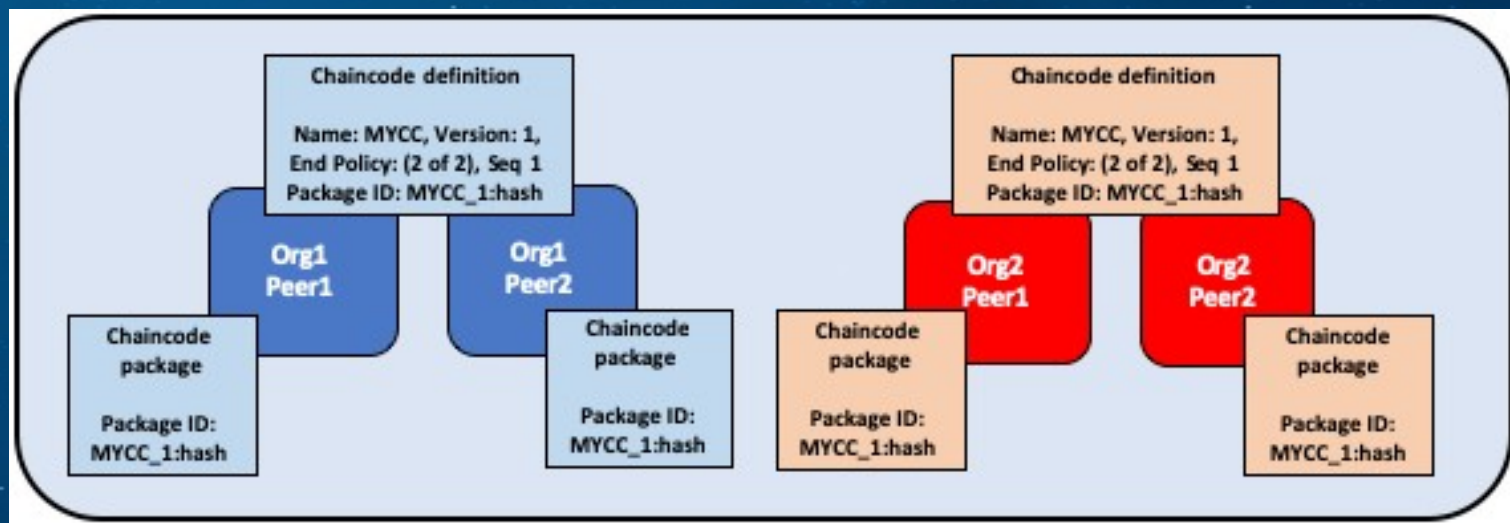


Packaging a smart contract



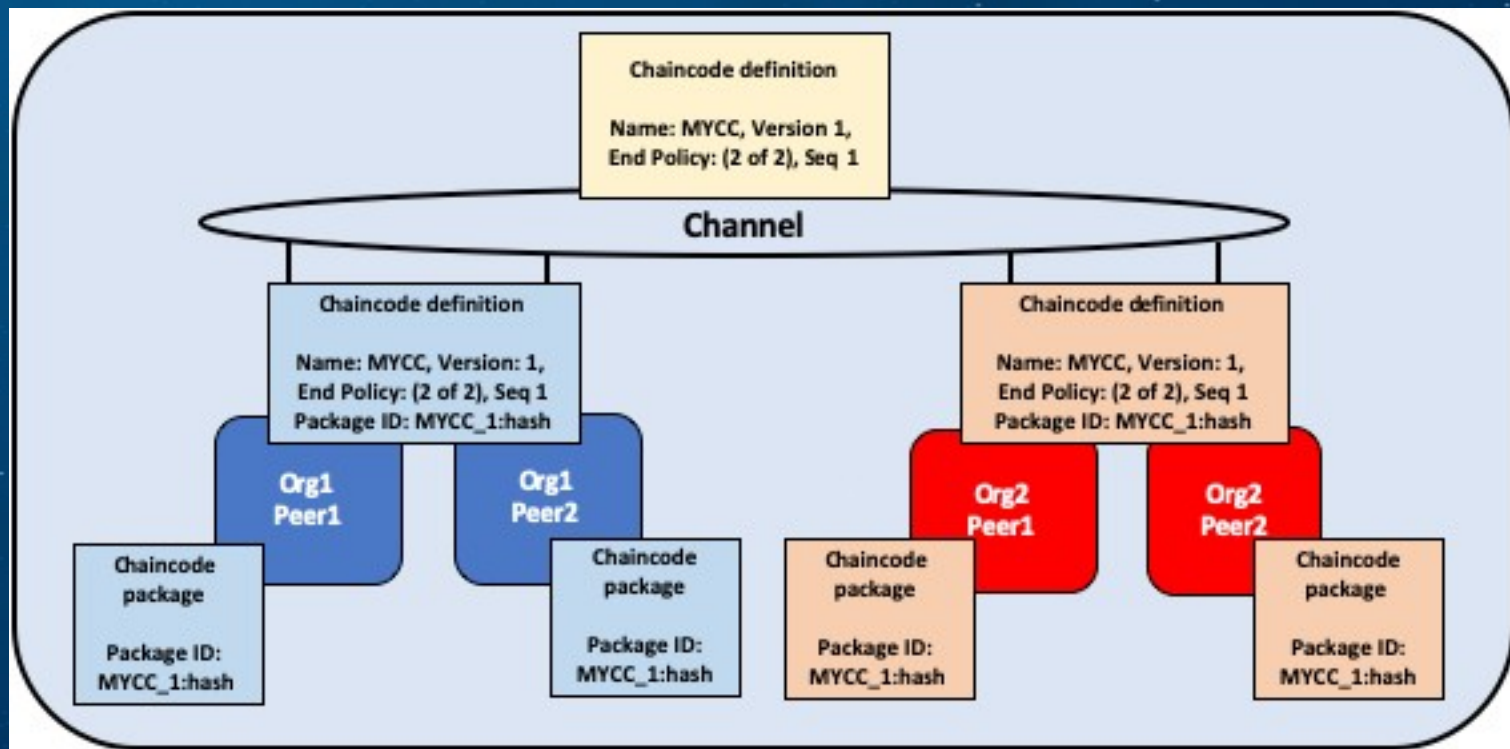


Installing a chaincode



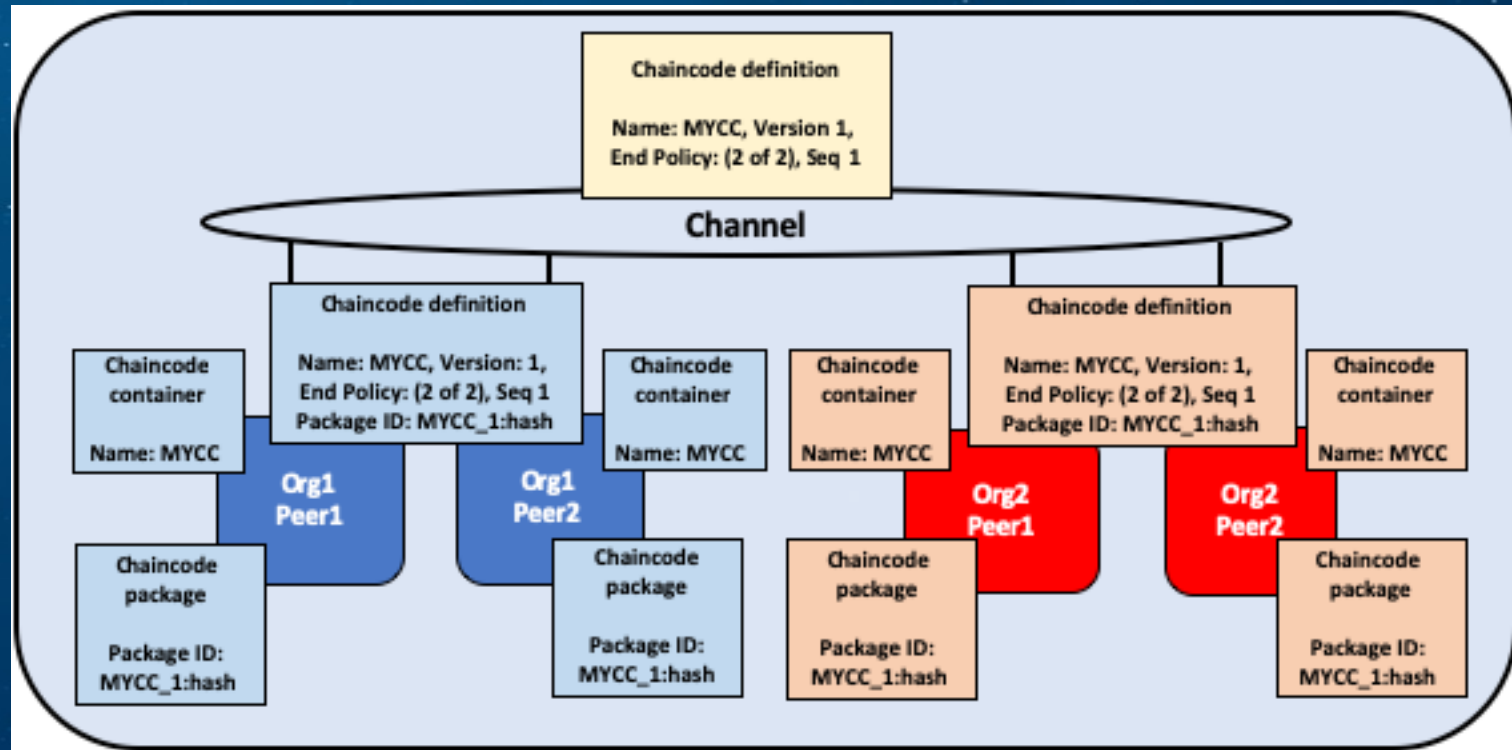


Approving a chaincode definition





Committing a chaincode definition





Chaincode access control

Chaincode can utilize the client certificate for access control decisions.

There are also extension APIs in Go (CID library) that extract information from the submitter's certificate used for access control decisions, whether that is based on client identity itself, or the org identity, or on a client identity attribute.

Ex : An asset may include the client's identity as one of its key/value pairs (e.g. asset owner), and only this client may be authorized to make updates to the asset.

Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

Membership

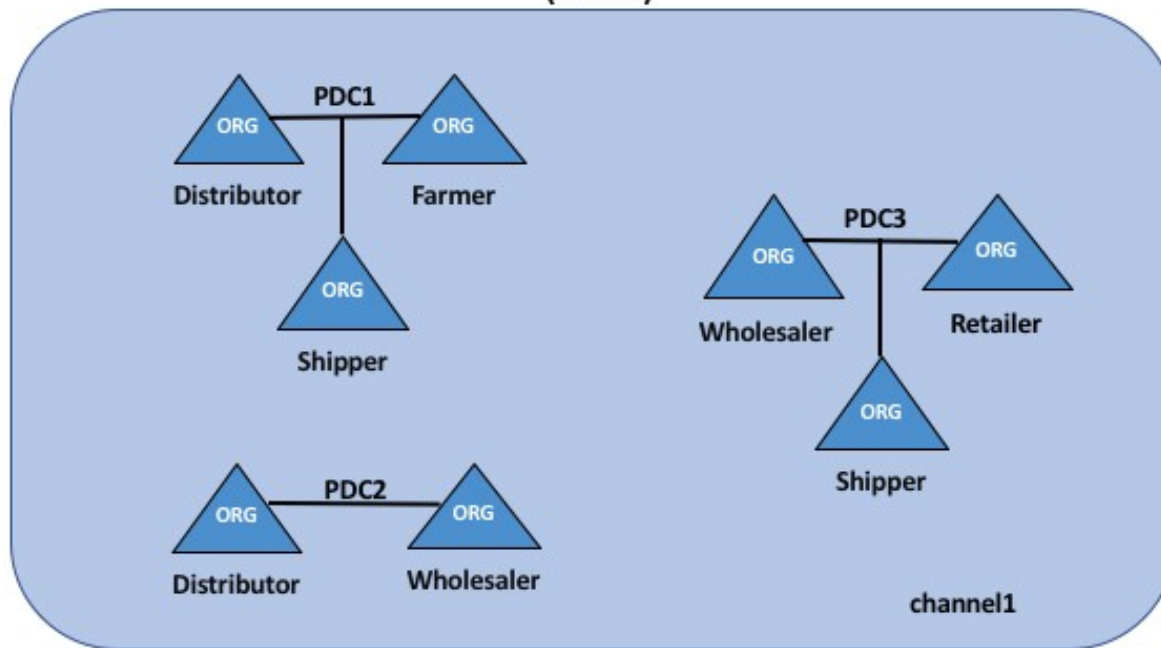
Chaincode lifecycle

Private data



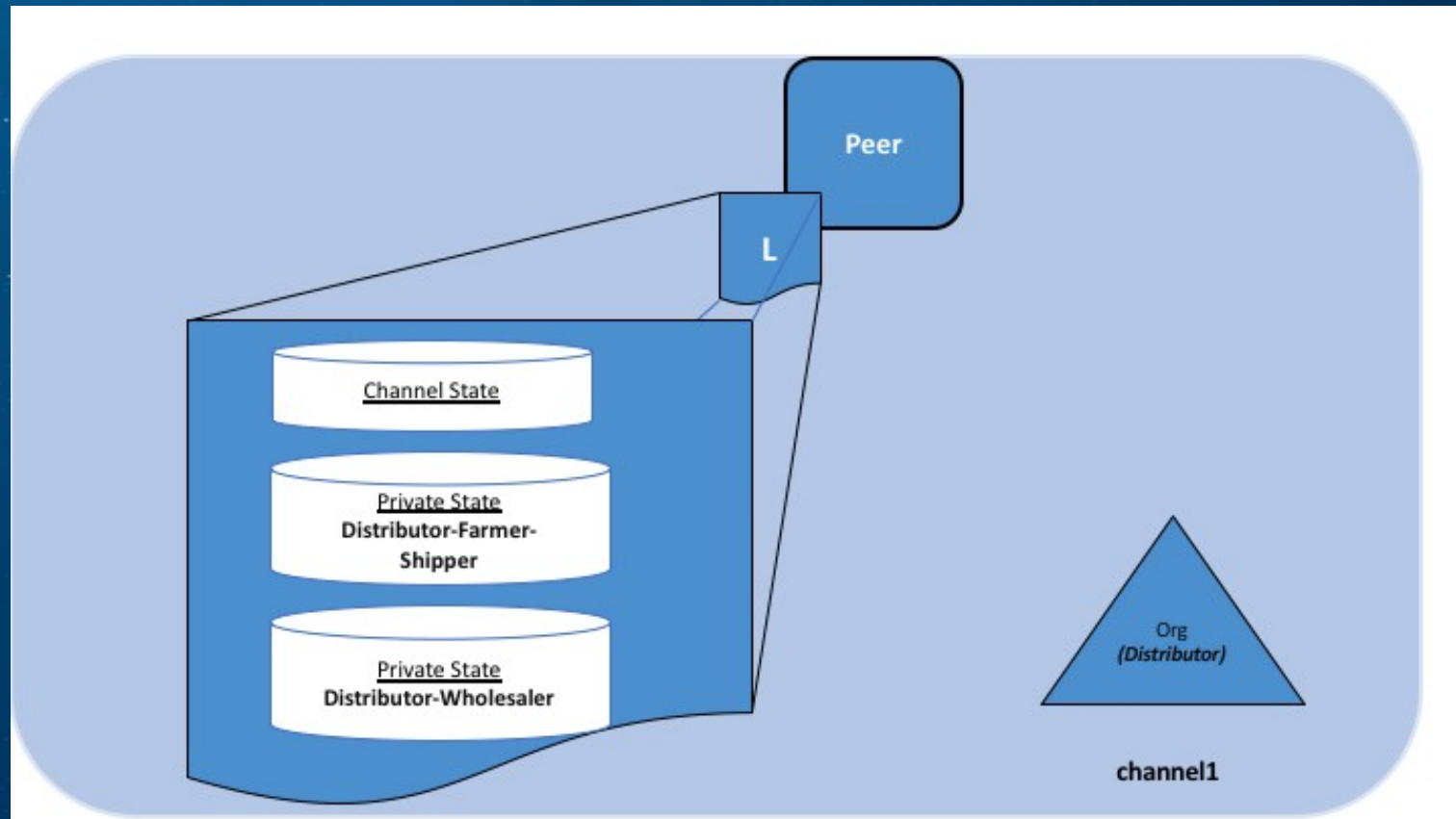
Private Data Scenario

Private data collections (PDC)





Private Data Collection



Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

Membership

Chaincode lifecycle

Private data



Chaicode encryption

Chaincode encryption is achieved by leveraging the entities extension which is a BCCSP wrapper with commodity factories and functions to perform cryptographic operations such as encryption and elliptic curve digital signatures.

Ex : to encrypt, the invoker of a chaincode passes in a cryptographic key via the transient field. The same key may then be used for subsequent query operations, allowing for proper decryption of the encrypted state values.

Fabric goals

Fabric architecture

Peers

Chaincode

Ledger

Transactions

Orderers

Decentralized App

Identities

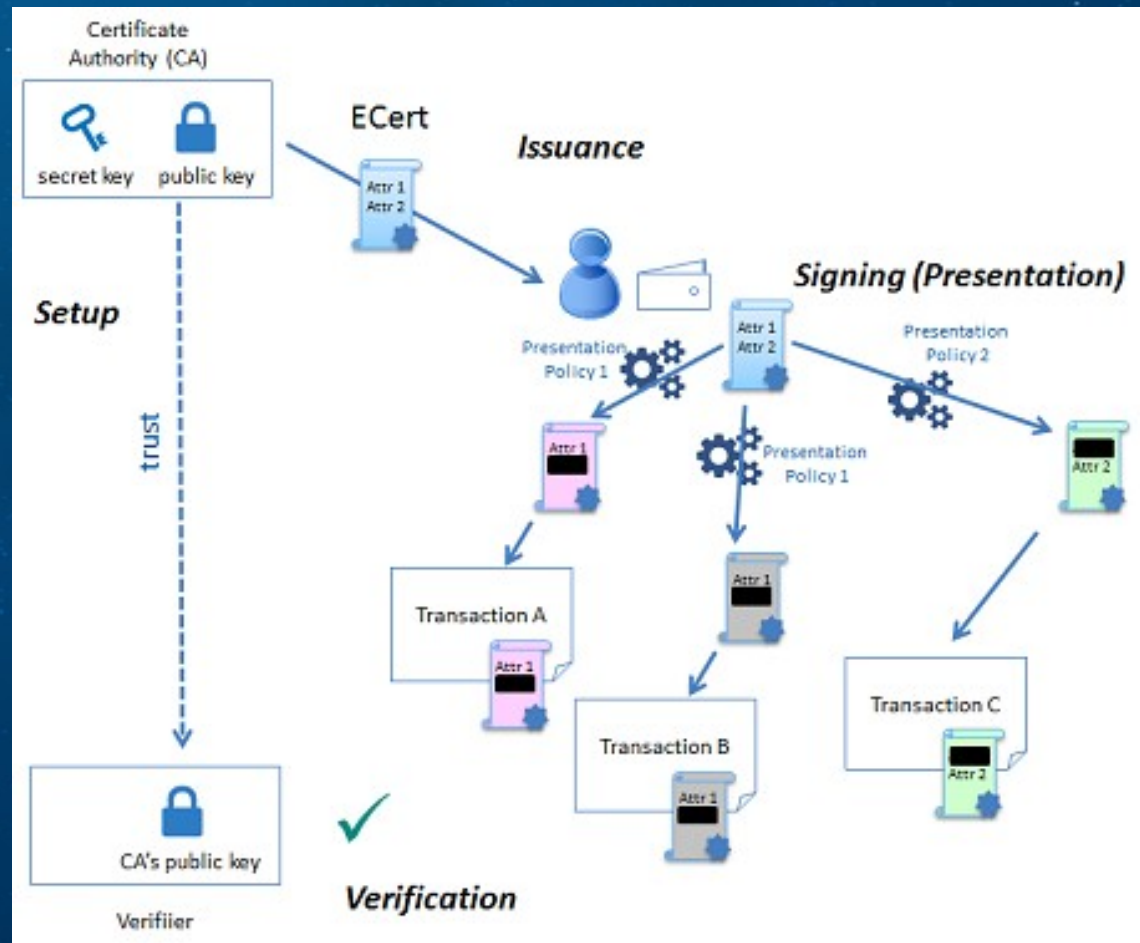
Membership

Chaincode lifecycle

Private data

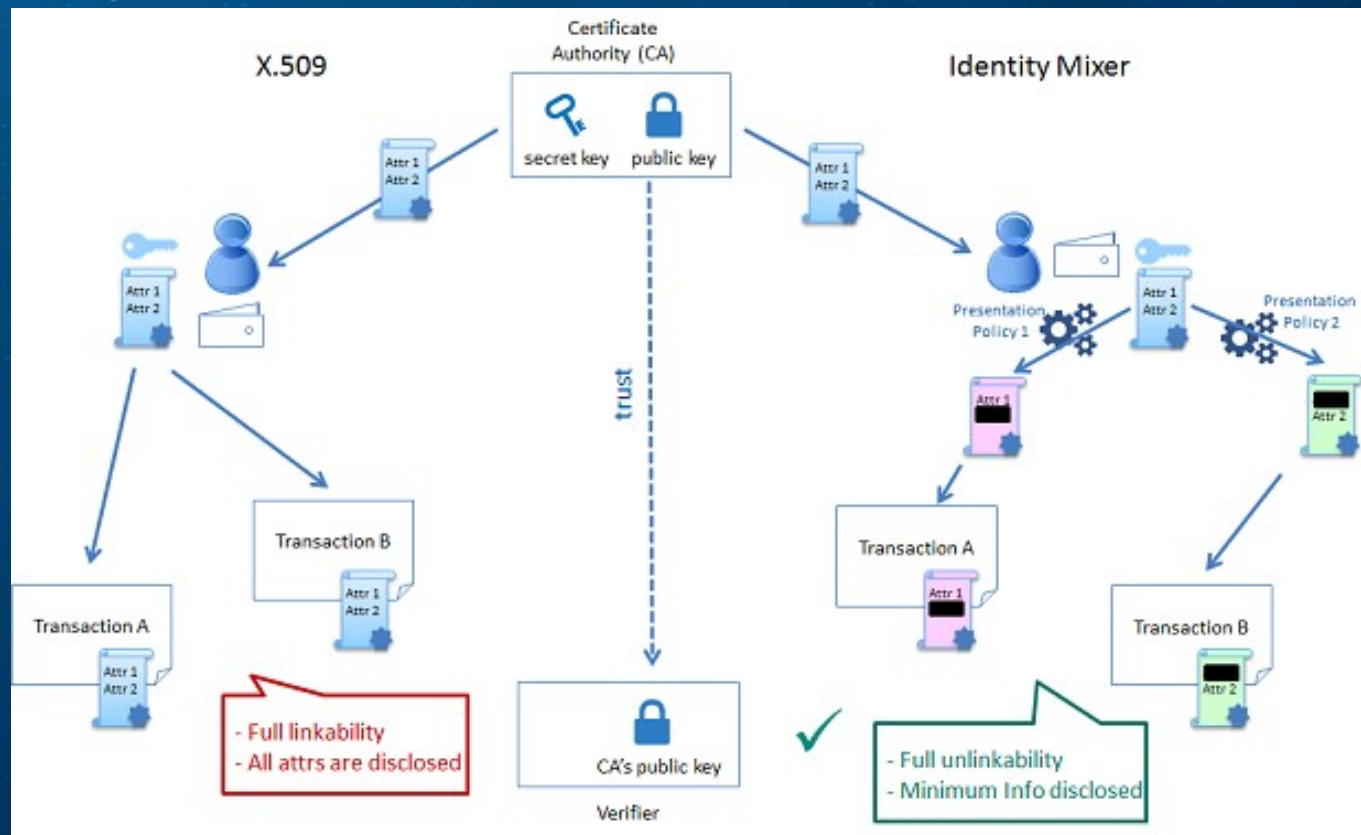


Identity Mixer



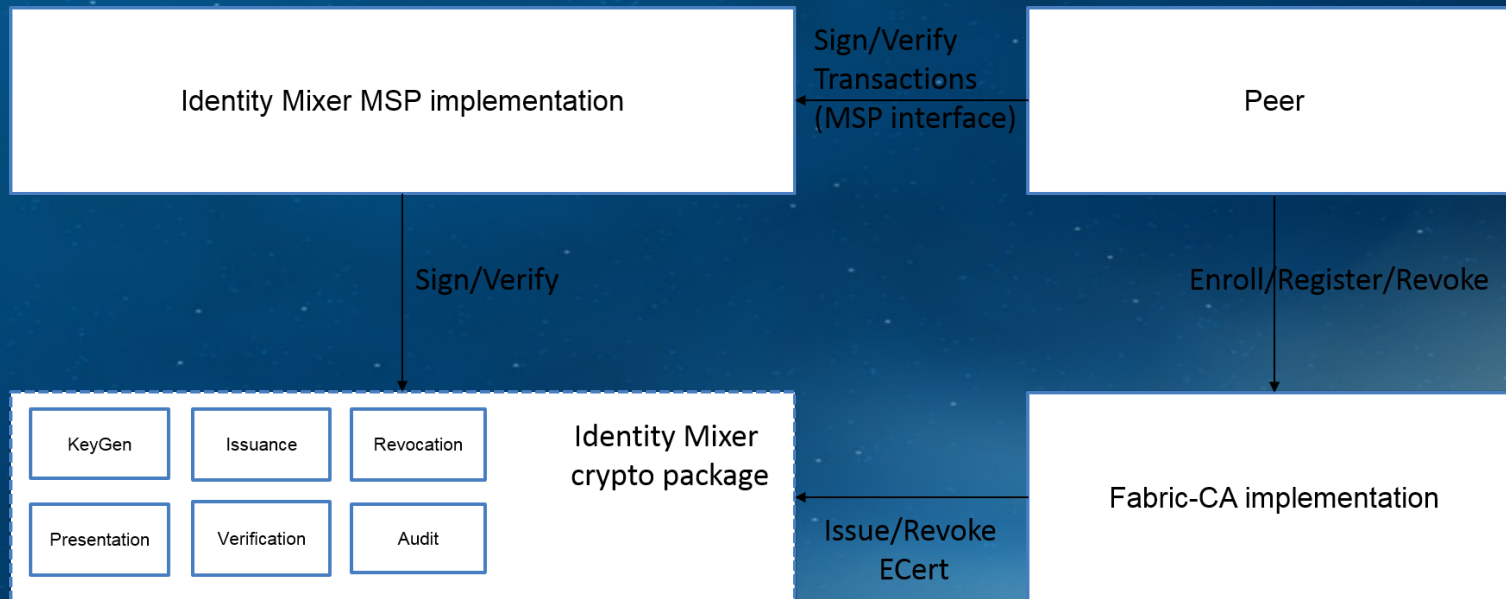


Identity Mixer



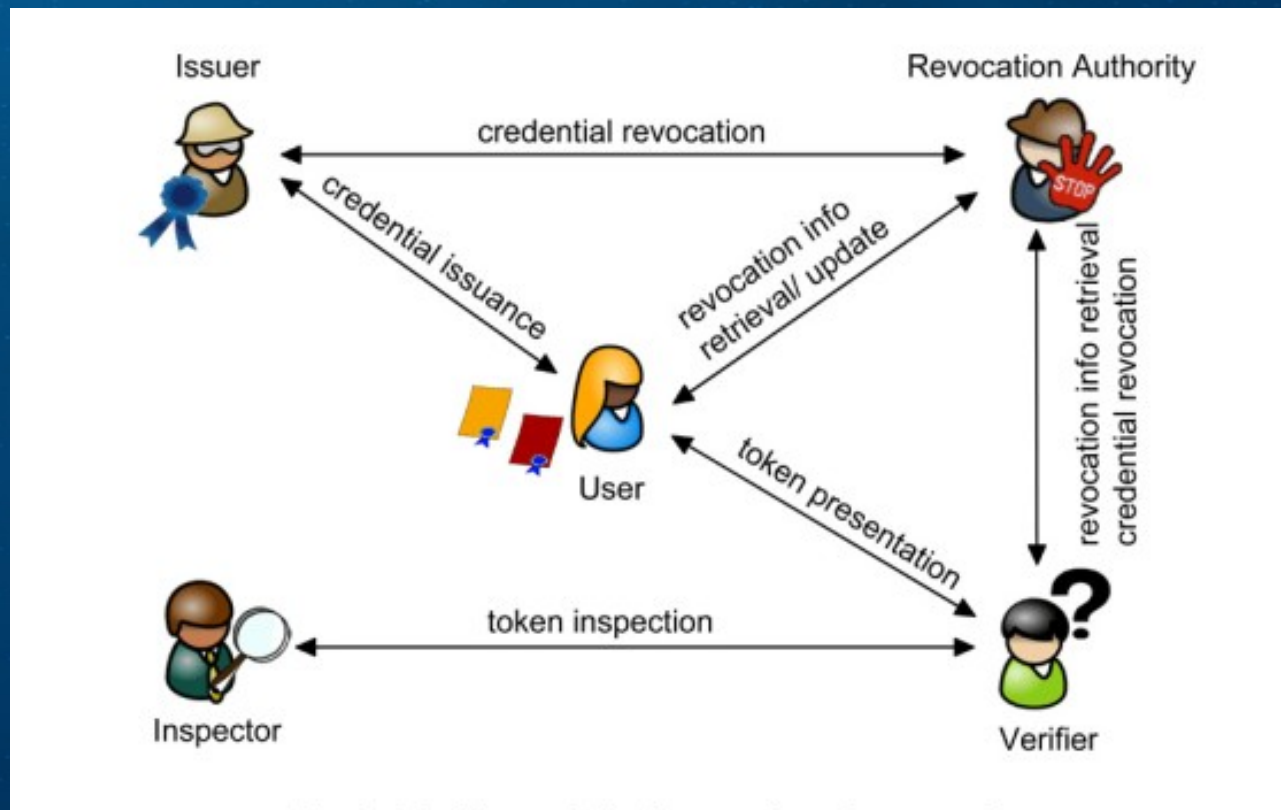


Identity Mixer





Identity Mixer



Camenisch J. et al. Concepts and Languages for Privacy-Preserving Attribute-Based Authentication, 2013



Identity Mixer

Credential : a certified container of attributes issued by an issuer to a user. Unlike traditional public-key certificates, a credential is not bound to a unique public key but just to a unique secret key, thanks to a suitable signature scheme and related protocols to prove knowledge of a signature (*)

Pseudonyms : are the different public keys that the user can generate for a given credential. They have the property of being unlinkable : given two different pseudonyms, one cannot tell whether they were generated from the same or from different secret keys.

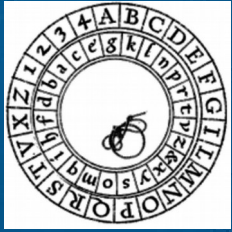
Scope-exclusive pseudonym : is unique given a user's secret key and a scope string (**).

Inspection : one or more credential attributes in a presentation token contains are encrypted under the public key of a trusted inspector. The verifier can check that the correct attributes were encrypted, but cannot see their actual values.

Presentation tokens : only reveal the attributes explicitly requested by the presentation policy, and no collusion of issuers and verifiers can link two presentation tokens. Presentation policies and tokens can also request and reveal predicates over one or more issued attributes.

(*) Camenisch J. and Lysyanskaya A.: A Signature Scheme with Efficient Protocols (2003)

(**) Technical Report RZ3730, IBM (2010)



`giovanni.schmid@icar.cnr.it`

Credits:

All images are from: <https://hyperledger-fabric.readthedocs.io>, except for those in slides 21, 42-44, which are not protected by copyright.