




# Hyperledger Fabric Security

Concepts & Components



# Introduction to Hyperledger Fabric

- Hyperledger Fabric is a permissioned blockchain system that allows permits unknown identities to be part of the network. The unknown identities have to register through the Membership Service Provider (MSP).
- For the maintenance of the identities of all the participating nodes (clients, ordering service nodes (OSNs) and peers) responsible is the membership service provider.
- MSP is one the most critical components of the platform, since it manages any type of access by issuing credentials in a form of cryptographic certificates that are used for authentication and authorization.

# Evolution of Hyperledger Fabric

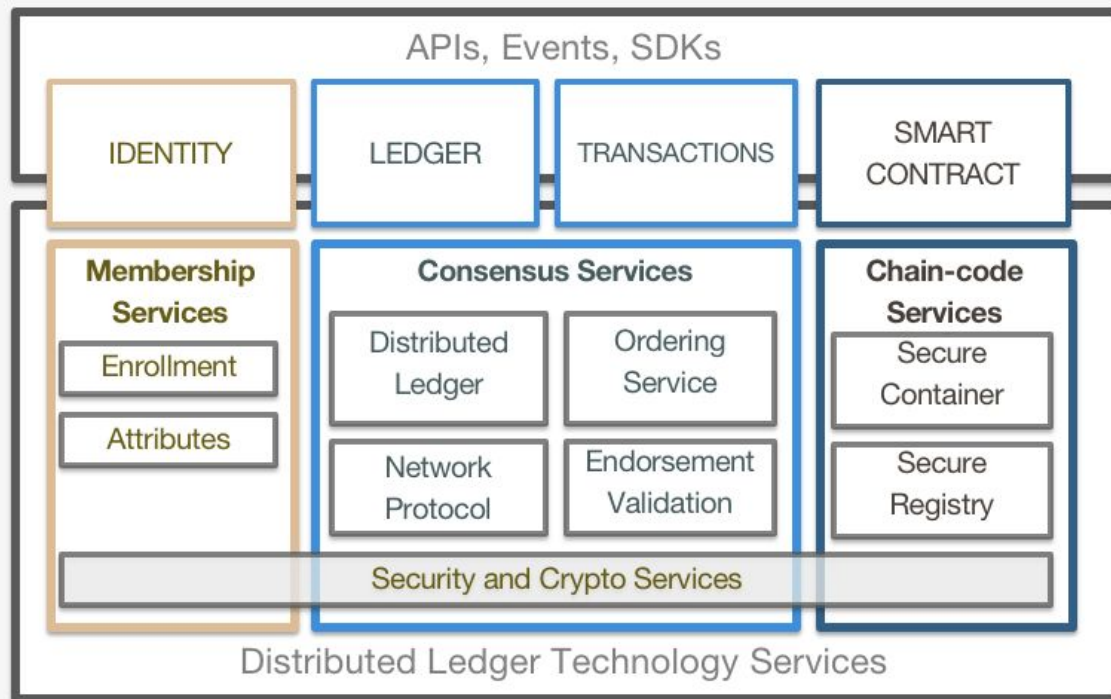
- In December 2015, the Linux Foundation and 30 initial companies set up a Hyperledger project to promote cross-industry blockchain technology and provide open source reference implementations for transparent, open, decentralized enterprise-level distributed ledger technology. By August 2018, the Hyperledger has more than 250 members, including Intel, Accenture, Huawei, JD, and other well-known enterprises.

# Hyperledger Projects in Blockchain Space



# Fabric Architecture

# Reference Architecture



## IDENTITY

Pluggable, Membership, Privacy and Auditability of transactions.

## LEDGER | TRANSACTIONS

Distributed transactional ledger whose state is updated by consensus of stakeholders

## SMART CONTRACT

“Programmable Ledger”, provide ability to run business logic against the blockchain (aka smart contract)

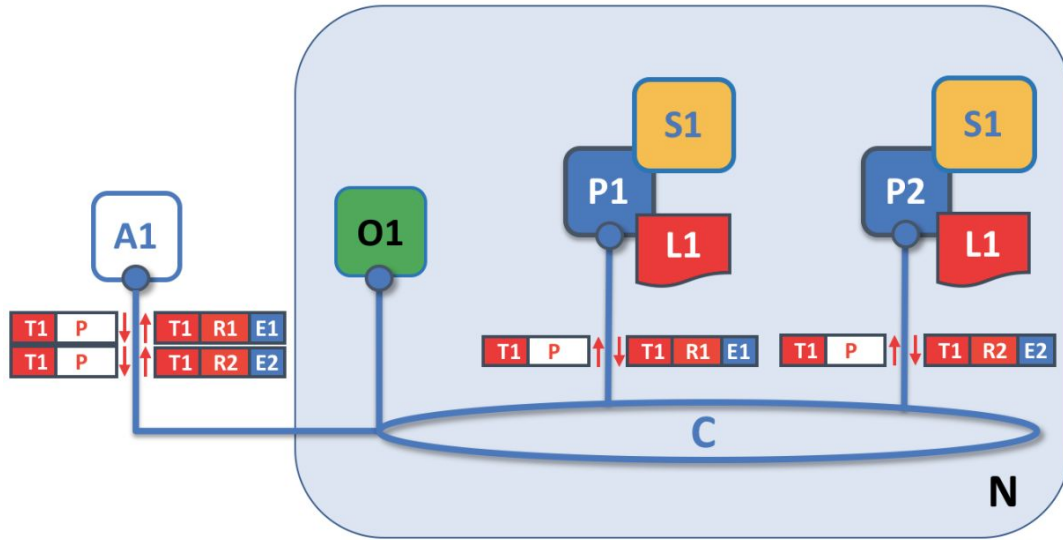
## APIs, Events, SDKs









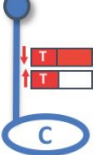

Multi-language native SDKs allow developers to write DLT apps

# Key Features of Hyperledger Fabric

- Assets: Enable the exchange of monetary value over the network
- Chaincode: Partitioned from transaction ordering, limiting the required levels of trust and verification across node types, and optimizing network scalability and performance
- Ledger Features: Encodes the entire transaction history for each channel, and includes SQL-like query capability Privacy through
- Channels: Enable multi-lateral transactions with the high degrees of privacy and confidentiality
- Security & Membership Services: In Permissioned membership participants know that all transactions can be detected and traced by authorized regulators and auditors
- Consensus: Allow network starters to choose a consensus mechanism that best represents the relationships that exist between participants

# Hyperledger Fabric Component View



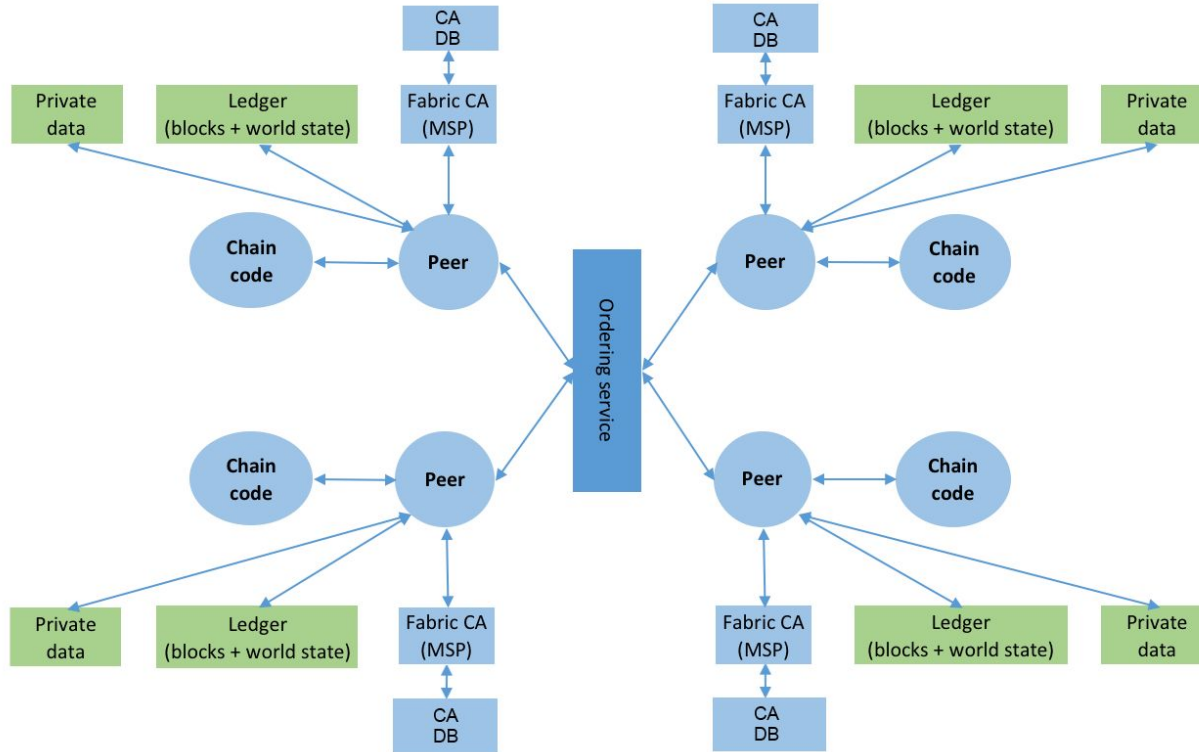
	Blockchain Network		Chaincode
	Channel		Orderer
	Peer		Ledger
	Transaction T proposal P		Transaction T1, response R2 endorsed with E2
	Ledger transaction T1 flows on channel C		Principal PA (P1,P2) communicates via channel C.



# Smart Contracts in Hyperledger Fabric

- In HF systems and applications, smart contracts are often referred to as chaincodes. Blockchain networks use chaincodes to constrain and regulate the reading or modification of key-value pairs and other state databases in order to accomplish complex business logic.
- The chaincode has its own execution logic, which performs the logical processing of the account book data according to the custom rules.
- A node supporting endorsement in the network installs and instantiates the chaincode on a node server so that business people can use the client with Fabric-SDK to communicate with the node service and realize the invocation of the chaincode.

# Hyperledger Fabric Application Architecture



# Architectural Approaches in Fabric

- Traditional blockchain platforms, permissionless and permissioned ones, follow a sequential execution style, whereby transactions on smart contracts are typically executed after consensus or entwined with it and where all participants execute all contracts.
- This order-execute architecture limits the scalability, requires sequential execution of transactions, and endorsement by all peers.
- Hyperledger Fabric introduces a novel approach that revamps the way blockchains cope with non-determinism and security issues such as resource exhaustion or performance attacks.
- Hyperledger Fabric uses a new execute-order-validate architecture, which lets transactions execute before the blockchain reaches consensus on their place in the chain

# Architectural Approaches in Fabric

Fabric pioneers a hybrid replication approach, combining passive and active replication in the Byzantine model.

Fabric replication is passive in the sense that every transaction is executed or endorsed by a subset of peers only, which allows for parallel execution and non-determinism.

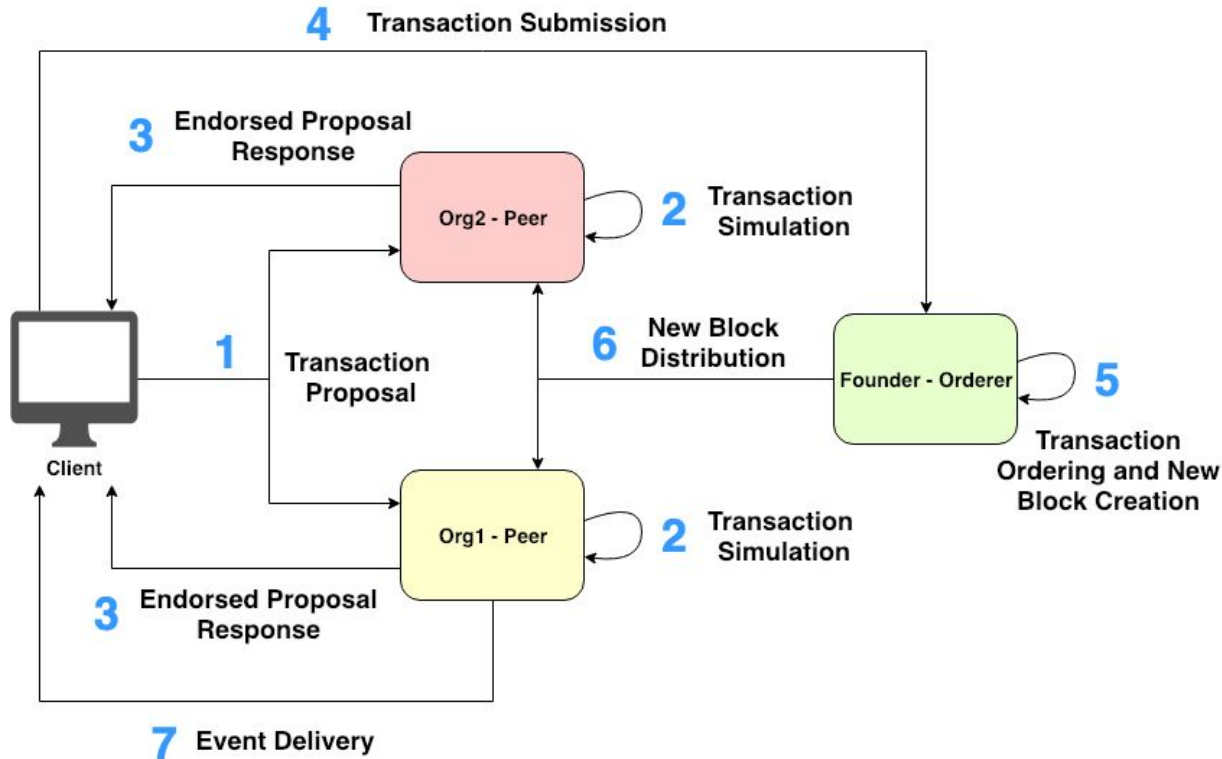
And because the effects of transactions on the ledger state are written only after a consensus is reached on their order, Fabric also uses active replication.

# Architectural Approaches in Fabric

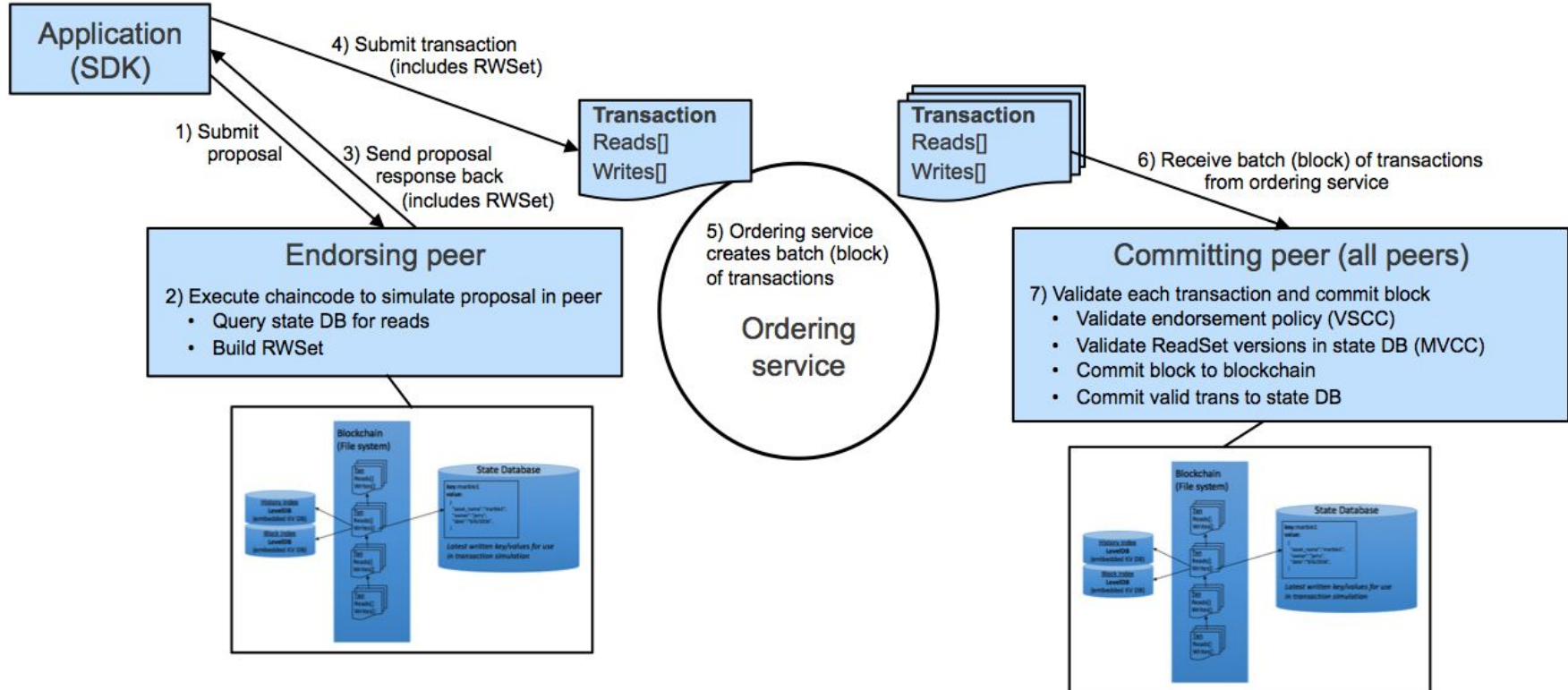
Essentially, this hybrid replication design supports a flexible endorsement policy in any given smart contract and simultaneously allows Fabric to respect application-specific trust assumptions according to the transaction endorsements.

The transactions don't have to be constructed all in one order, as long as they are consistent with each other and come together at the right time.

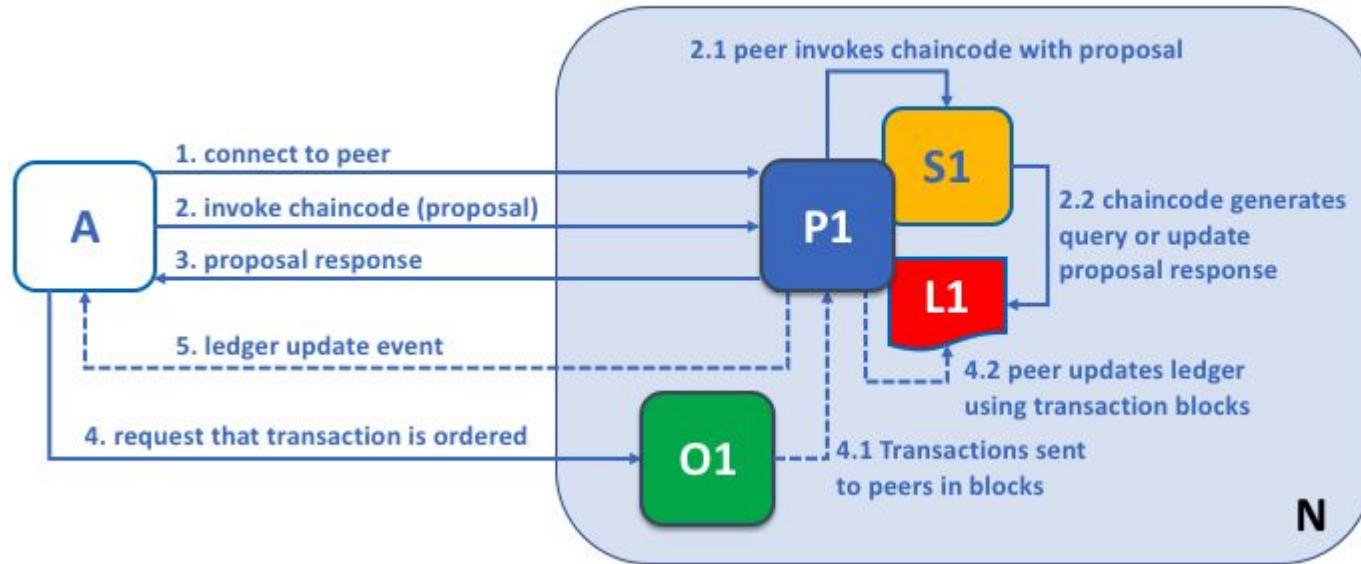
# Transaction Lifecycle in Hyperledger Fabric



# Deep Dive on Transaction Lifecycle



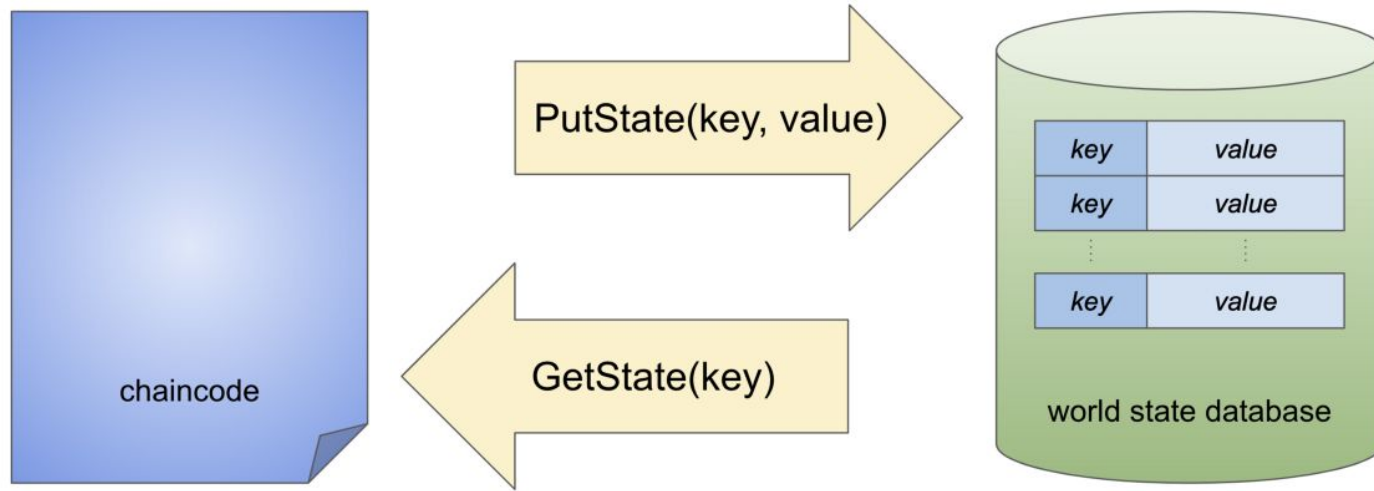
# Chaincode Execution Approach



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



# Chaincode and State Database



# Data Architecture : World State and Private Data

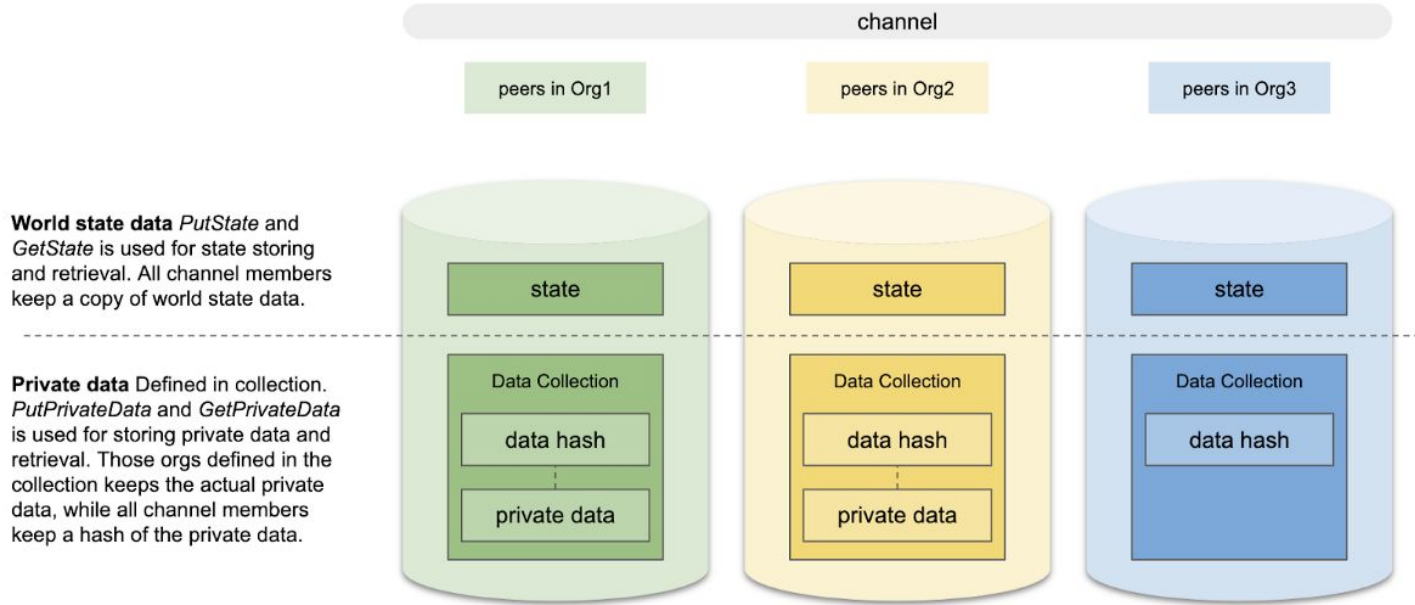
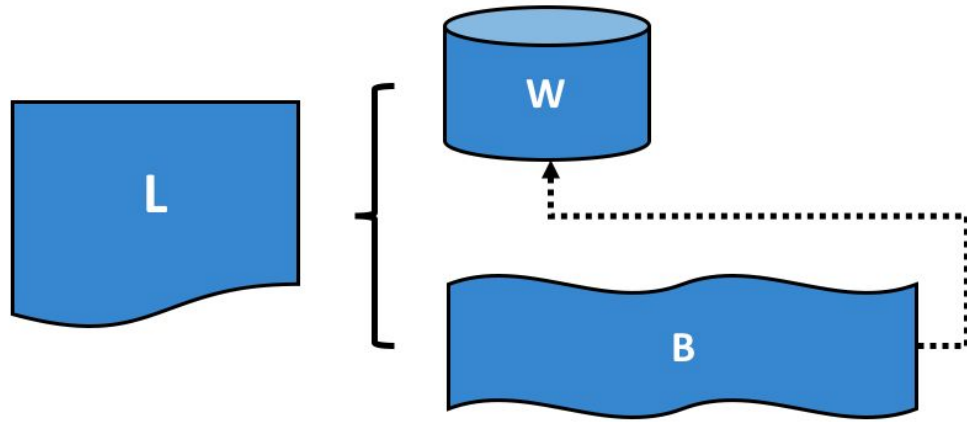



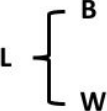
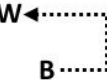


Illustration of private data. This shows the world state database for each peer in Org1, Org2 and Org3. In this illustration, all organizations join a channel, and the data collection is defined for Org1 and Org2 only.


# Ledger, World State and Blockchain



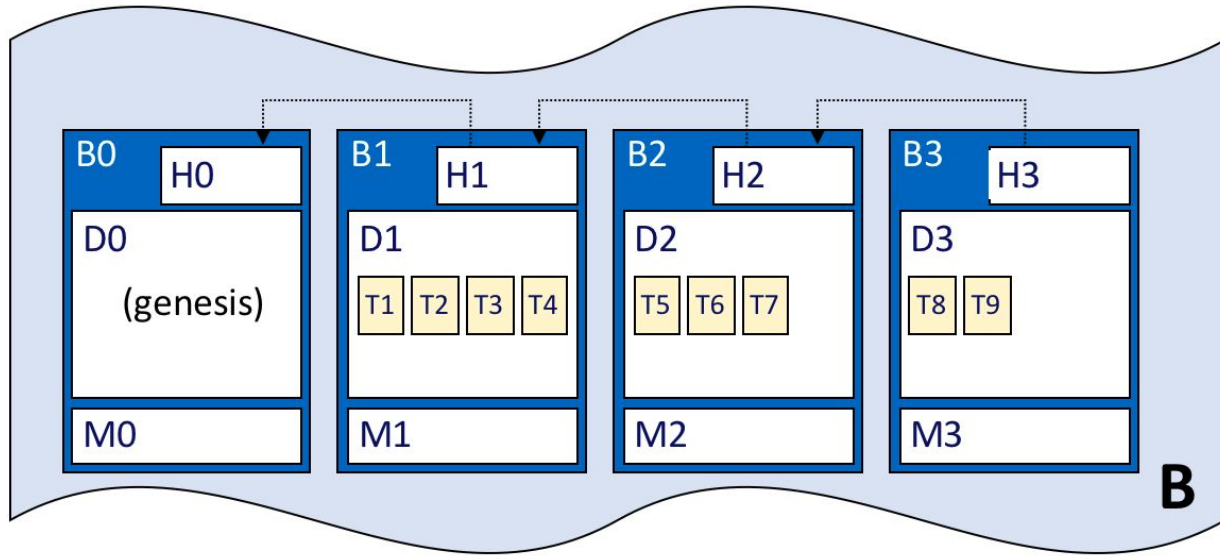
	Ledger
	World State
	Blockchain
	L comprises B and W
	B determines W



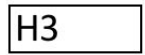
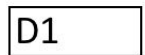

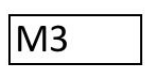
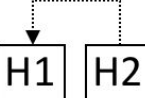
# A Deep Dive into the State Database



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

# Block Data Structure in Hyperledger

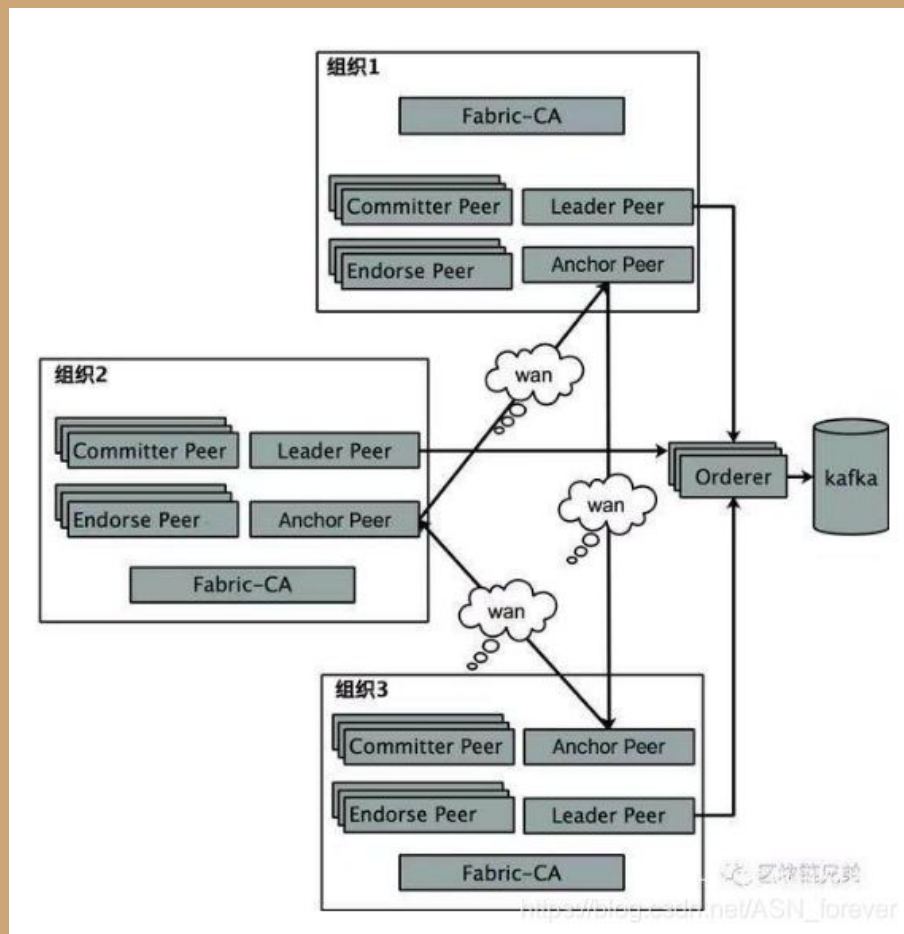


 B	Blockchain
 B1	Block
 H3	Block header
 D1	Block data
 T5	Transaction
 M3	Block metadata
 H1 H2	H2 is chained to H1

# A Deep Dive into the State Database

# Type of Peers in Hyperledger Fabric

Endorsing Peer  
Anchor Peer  
Leader Peer  
Committer Peer



# Anchor Peers and Leader Peers

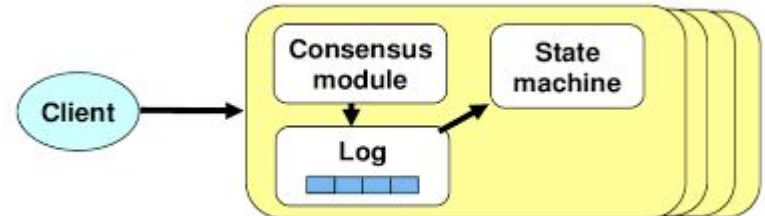
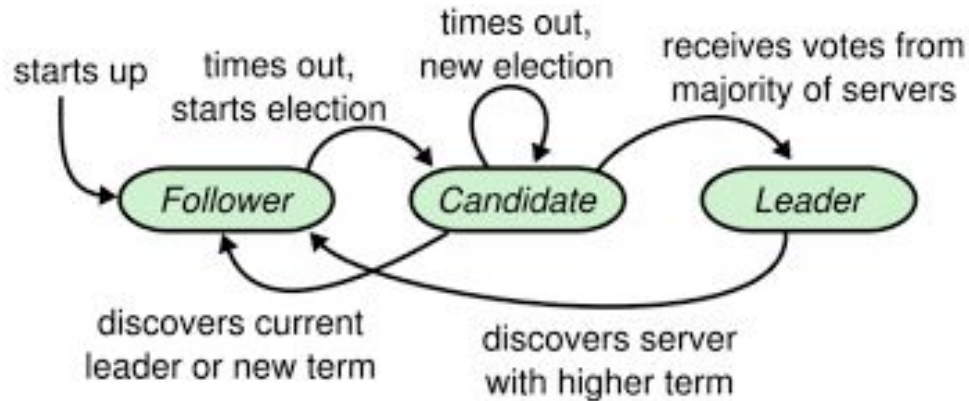
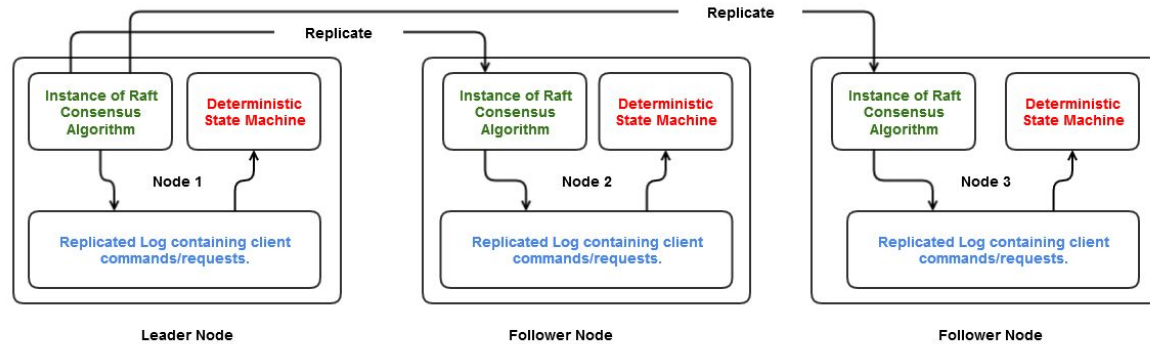
- Anchor Peer
  - It's a peer node on a channel that all other peers can discover and communicate with. Therefore, every participating organization in a channel has an anchor peer. Peers belonging to an organization can query this peer to discover all peers belonging to other organizations in the channel.
- Leader Peer
  - When an ordering service node must send a block to the peers in the channel, it sends the block to each of the leading peer associated with the organizations. The leading peer in turn distributes this block to other peers belonging to the same organization. They can be configured in two ways.
    - Static Leader
    - Dynamic Leader



# RAFT Consensus in Hyperledger Fabric

- Raft: The Raft consensus protocol is based on the “leader-election” model to establish consensus by electing a leading node to acquire the incoming entries from the clients and replicate them.
- To provide strong leader and coherency, the protocol is separated into three phases, i.e. the leader election, the log replication and the safety. The time in Raft proceeds in arbitrary time periods, called “terms”, with each term to be defined by an increasing number.
- The nodes in Raft are hierarchically ranked in different states, with each node to be either a leader, a follower or a candidate. The leader is the principal entity of the protocol and it is elected per channel, with the task to interact with the clients and then replicate its entries to its synchronized followers.
- Therefore, in order to achieve the best possible synchronization, it sends systematic heartbeats to its followers and even if the network suspects that the leader has crashed, at least one of its followers will detect this divergence, cast a vote to the network and attempt to take its place.
- Some nodes might compete to win the election by seeking votes from other nodes. Therefore, these nodes are considered as “candidates”.

# RAFT Consensus Algorithm



# Hyperledger Fabric Security in Context

- As a permissioned blockchain, Hyperledger Fabric network threats differ from popular permissionless chains.
- For example, 51% attacks and network partitioning attacks are not as significant of a threat on permissioned networks because users are known, their activities can be monitored, and access is managed by access control lists.
- Some of these attacks are common to all distributed systems like Denial of Service (DoS) or consensus manipulation. Other attacks target specific components in a Hyperledger Fabric network, such as the Membership Service Provider (MSP).

# Hyperledger Fabric Security Model

Hyperledger Fabric is a permissioned blockchain where each component and actor has an identity, and policies define access control and governance.

- Identity Management
- Membership Service Providers
- Policies
  - Channel Policies
  - Chaincode Lifecycle Policies
  - Chaincode Endorsement Policies
- Transport Layer Security
- Hardware Security Modules

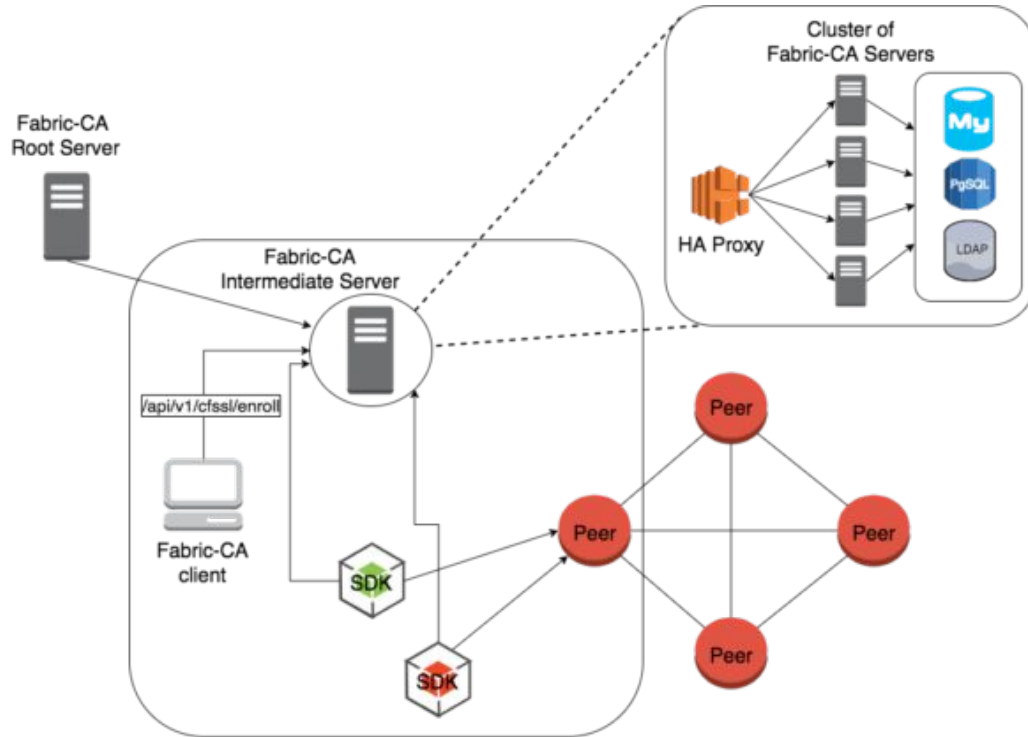
# Functional Security in Fabric

- Connecting TLS encryption and proper certificate handling are the most important aspects of functional security. Blockchain data is secure by design, however, the functional part of this security relies on proper [configuration of Hyperledger certificate authority](#) (CA) with proper key management.
- We can also setup Attribute-Based Access Control, which allows [smart contracts](#) to operate on [specific client attributes](#). This, along with enabling [TLS client authentication on peer nodes](#) sets the overall trust level in the whole network reasonably high.

# Functional Security in Fabric

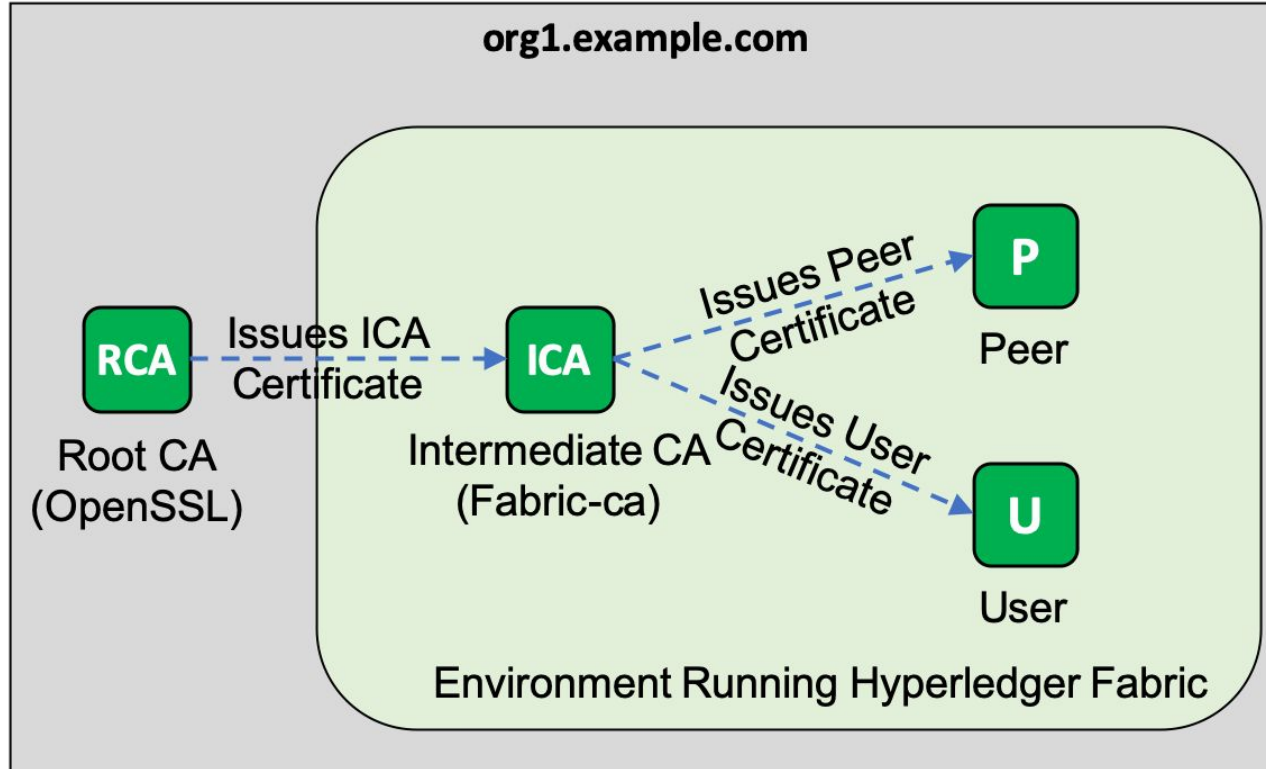
- Apart from the network level, there's still the host level, on which malicious actors can steal data unless it's under encryption.
- We can use either Hyperledger Fabric native encryption or on the filesystem level such as LUKS or on a [cloud provider level such as the AWS Key Management Service.](#)

# Functional Security in Fabric



Blockchain

# Root CA and Intermediate CA





# Certificate Authority Capabilities

- The Hyperledger Fabric CA is a Certificate Authority (CA) for Hyperledger Fabric. It provides features such as:
  - registration of identities, or connects to LDAP as the user registry
  - issuance of Enrollment Certificates (ECerts)
  - certificate renewal and revocation
- Hyperledger Fabric CA consists of both a server and a client component
- The Fabric CA server should always be started with TLS enabled.
- Otherwise, server vulnerable to an attacker with access to network traffic.

# Identity Management in Hyperledger Fabric

- The different actors in a blockchain network include peers, orderers, client applications, etc. have distinct identity.
- These identities really matter because they determine the exact permissions over resources that actors have in a blockchain network. Hyperledger Fabric uses certain properties in an actor's identity to determine permissions, and it gives them a special name – a principal.
- Each of these actors — active elements inside or outside a network able to consume services — has a digital identity encapsulated in an X.509 digital certificate issued by a Certificate Authority (CA).

# Verifiable Identity from Membership Service Provider

- For an identity to be verifiable, it must come from a trusted authority. A membership service provider (MSP) is that trusted authority in Fabric.
- More specifically, an MSP is a component that defines the rules that govern the valid identities for an organization.
- A Hyperledger Fabric channel defines a set of organization MSPs as members.
- The default MSP implementation in Fabric uses X.509 certificates issued by a Certificate Authority (CA) as identities, adopting a traditional Public Key Infrastructure (PKI) hierarchical model.

# Importance of Policies in Hyperledger Fabric

- In Hyperledger Fabric, policies are the mechanism for infrastructure management. Fabric policies represent how members come to agreement on accepting or rejecting changes to the network, a channel, or a smart contract.
- Policies are agreed to by the channel members when the channel is originally configured, but they can also be modified as the channel evolves.
- For example, they describe the criteria for adding or removing members from a channel, change how blocks are formed, or specify the number of organizations required to endorse a smart contract.

# Attack Vectors

# Denial of Service Attacks

- DoS attacks disrupt the network's availability and are a threat to any distributed system.
- Many different attacks can result in denial of service, which makes it difficult to proactively prevent.
- This risk can be mitigated by collecting performance metrics, such as transaction throughput and latency, to detect compromised availability early on.

# Consensus Manipulation

- Attacks on the network consensus include DoS and transaction reordering attacks.
- Hyperledger Fabric currently only utilizes Crash Fault Tolerant (CFT) consensus algorithms, meaning it cannot tolerate any malicious actors.
- There is ongoing work on Byzantine Fault Tolerant (BFT) algorithms, which will be able to tolerate up to  $\frac{1}{3}$  of the network being malicious.
- Regardless of the consensus algorithm used, early detection of malicious behavior can mitigate this threat.
- Logging threat indicators, such as leadership elections and transaction latencies, is critical for detection.

# Compromise of Membership Service Provider

A compromised MSP can be a significant Fabric-specific threat.

The MSP is able to modify access control to the network and, if malicious, could deny service and perform sybil attacks. The MSP may be compromised by a rogue insider or through private key theft, which may only be detectable after exploitation.

To mitigate this risk, it is important to follow best practices with key management. Logging MSP actions, such as certificate creation and revocation, can help detect malicious behavior in case of compromise. Alerting based on that logging results in early identification and remediation.



# Chain Code Risks

- Random value
- Timestamp
- Iteration on map object
- Calling external API
- File access
- Global variable
- External library
- System commands
- Goroutine
- Range query risk
- Read your write

# ChainCode Vulnerabilities

- Object reification: The value of the variables are handled through a pointer, which is an address of memory. Therefore, using reified object addresses might cause non-determinism.
- System Timestamp: It is difficult to ensure that the timestamps are run concurrently in each peer.
- The global state variables: Global state variables that are not stored to the ledger, might change innately and cause inconsistencies to the endorsing results.
- Map structure iteration: Due to the hidden implementation details of the Go programming language, when an iteration with map structure is used, non-determinism may arise and the order of key values might be different.

# ChainCode Vulnerabilities

- Concurrency: If multiple transactions are executed concurrently and under high load, a possible change at the keys' versions might lead to key collisions and double spending.
- If a transaction has passed from the endorsing phase and its version key has changed before it reaches the validation phase not only a program error will occur but this action might also allow another (possible double spent) transaction to be included to the ledger.
- The non-determinism that ascends from peripheral resources: Some accessing resources, such as web services, external libraries, external files and system command executions; can corrupt the code and return different endorsing results among the endorsing peers.

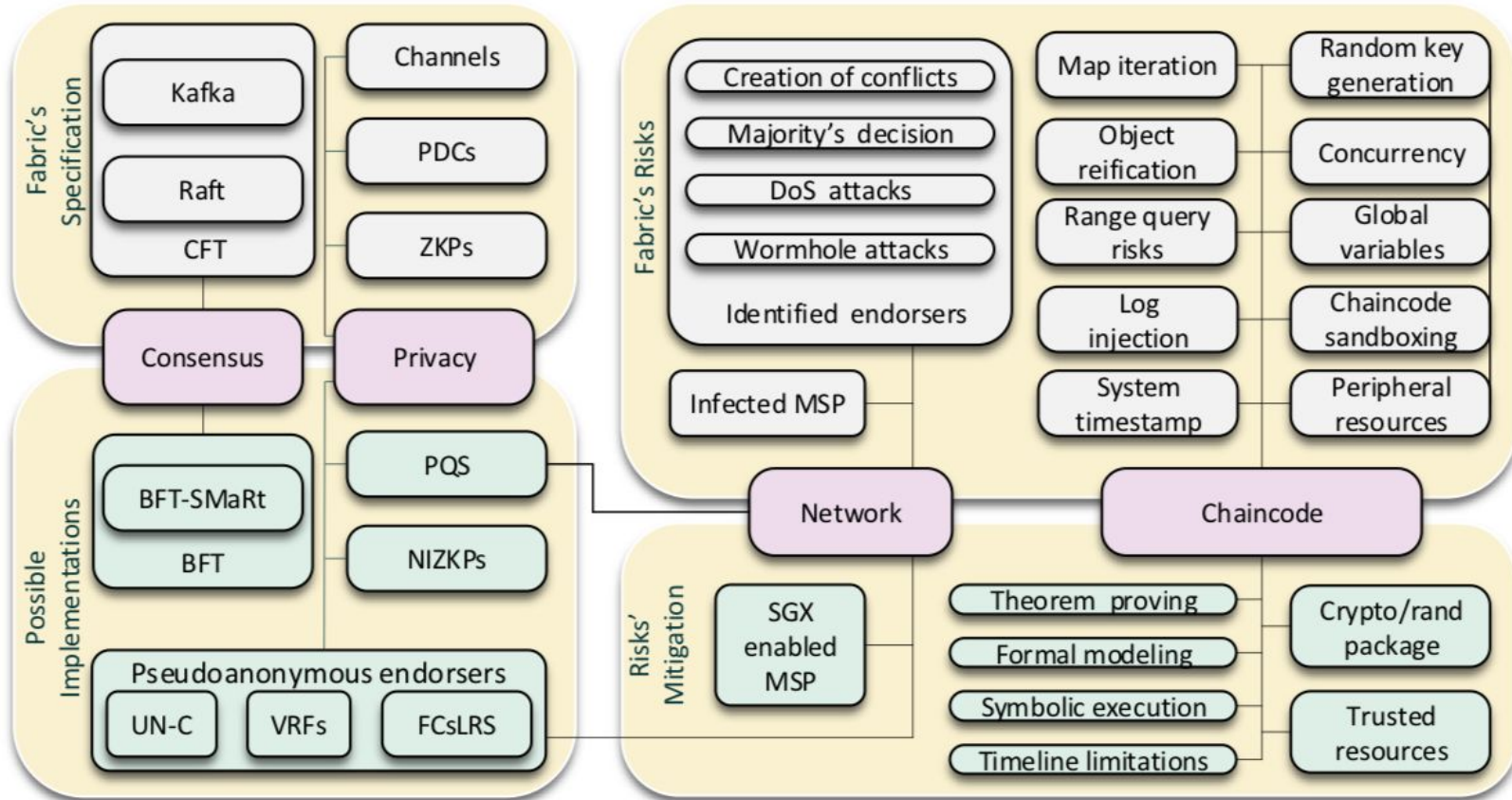
# ChainCode Vulnerabilities

- Range query risks: Queries methods to access the Fabric's state databases and obtain private data, (e.g. the history or the state of a key); are not executed again in the validation phase and can lead to phantom reads, in which the dirty data cannot be detected.
- Chaincode sandboxing: Although, Fabric's chaincode is executed in an isolated docker container and provides sufficient privileges, it can be exploited in a malicious way, such as to execute port scanning, identify and exploit vulnerable peers, install malicious software and execute attacks.
- Log injection: Any corruption of the log messages can possible avert them from being executed automatically and allow the attacker to view the processed logs.



# Risk Mitigation Methods





# Smart Contract Security Measures

- Timeline limitations of the chaincode execution can be employed;
- The chaincode execution can be performed with privileges other than root;
- GO's crypto/rand package can be used to produce a cryptographically secure random seed for the keys and minimize the risk of double spending;
- Any changes at the keys' versions can increase the possibility of double spending and thus, should be avoided;
- Only trusted services peripheral of the platform should be accessed;
- Chaincode input arguments should be checked for escape characters;



# Privacy Protection in Fabric





# Privacy Protection Mechanism in Fabric

- The privacy protection measures of Hyperledger Fabric include the following four aspects:
  - Firstly, asymmetric cryptography and zero-knowledge proof separate the transaction data from on-chain records, protecting privacy from the underlying algorithm.
  - Secondly, the digital certificate management service guarantees the legitimacy of the organization on the blockchain.
  - Thirdly, the design of multi-channel separates the information between different channels.
  - Finally, privacy data collection further satisfies the need for the isolation of privacy data between different organizations within the same channel.

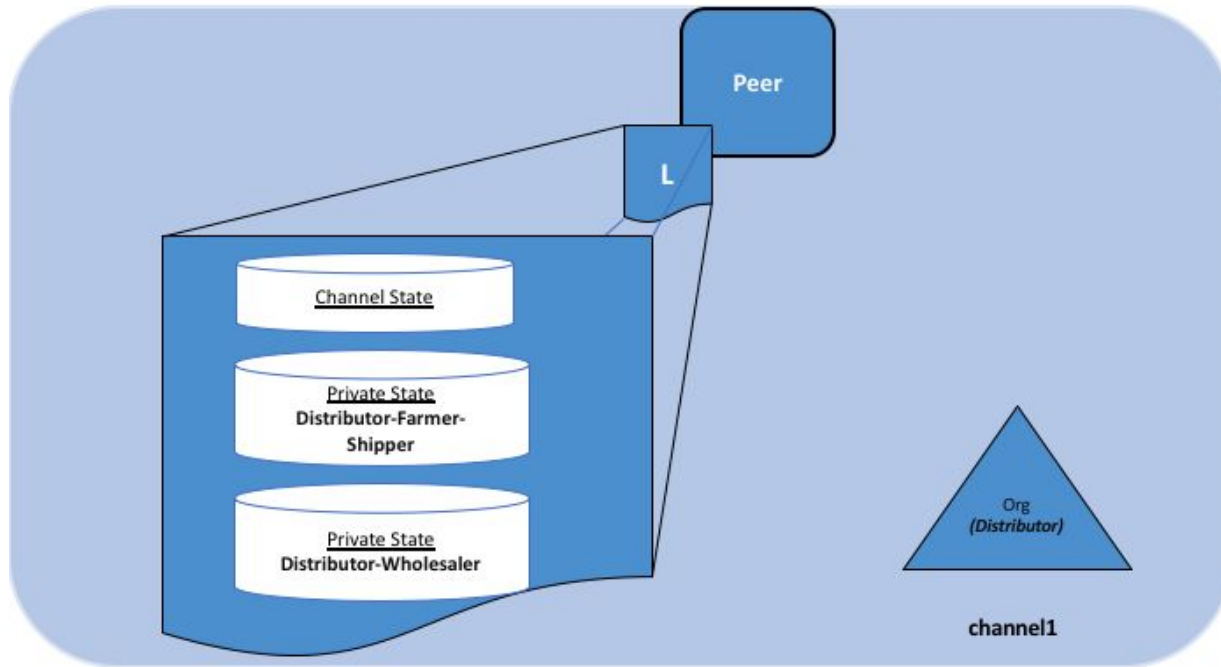
# Channels in Hyperledger Fabric

- “A channel is like a virtual blockchain network that sits on top of a physical blockchain network with its own access rules. Channels employ their own transaction ordering mechanism and thus provide scalability, ultimately allowing for effective ordering and partition of huge amounts of data.”
- Channels in Hyperledger Fabric are configured with access policies that govern access to the channel’s resources (chaincodes, transactions, and ledger state), thus preserving the privacy and confidentiality of information exclusively within the nodes that are in the channel.
- The channel is dedicated to the blockchain privacy protection, allowing the data on the channel to be isolated separately. The peer on the same channel shares a ledger, the transaction peer needs to obtain the recognition of the channel before it can join the channel and transact with others.

# Channels and Private Data Collection

- Creating separate channels in each of these cases creates additional administrative overhead (maintaining chaincode versions, policies, MSPs, etc), and doesn't allow for use cases in which you want all channel participants to see a transaction while keeping a portion of the data private.
- The PDC (private data collection) is a collection of organizations that are authorized to store private data on a channel, and the data stored includes:
  - (1) Private data, which implements peer-to-peer communication between authorized organizations through the Gossip protocol. The privacy data is stored in the peer's private database.
  - (2) The hash value of private data. For private data, the peers on the channel use the hash value of the private data when sorting and writing the endorsement, as evidence of the existence of the transaction and for state validation and auditing.

# Private Data and Channel Structure



# Channel v/s Private Data Collections

- Use channels when entire transactions (and ledgers) must be kept confidential within a set of organizations that are members of the channel.
- Use collections when transactions (and ledgers) must be shared among a set of organizations, but when only a subset of those organizations should have access to some (or all) of the data within a transaction.
- Additionally, since private data is disseminated peer-to-peer rather than via blocks, use private data collections when transaction data must be kept confidential from ordering service nodes.

# Transaction Flow with Private Data

- The client application submits a proposal request to invoke a chaincode function (reading or writing private data) to endorsing peers
- The endorsing peers simulate the transaction and store the private data in a transient data store
- The endorsing peer sends the proposal response back to the client. The proposal response includes the endorsed read/write set, which includes public data, as well as a hash of any private data keys and values.
- The client application submits the transaction (which includes the proposal response with the private data hashes) to the ordering service.

# Transaction Flow with Private Data

At block commit time, authorized peers use the collection policy to determine if they are authorized to have access to the private data.

If they do, they will first check their local transient data store to determine if they have already received the private data at chaincode endorsement time.

If not, they will attempt to pull the private data from another authorized peer.

Then they will validate the private data against the hashes in the public block and commit the transaction and the block.

Upon validation/commit, the private data is moved to their copy of the private state database and private writeset storage.

The private data is then deleted from the transient data store.

# Private Data Possibilities

- The database storing the private data is updated alongside the public ledger as transactions containing references to private data are committed. In fact, the hashes on the public ledger serve as verifiable proof of the data.
- Private transactions can be combined with anonymous client authentication to avoid leaking the connection between the identity of the transaction's creator and the ledger stored (hashed) data.





# References



# References

- <https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/>
- <https://www.hyperledger.org/blog/2021/11/18/hyperledger-fabric-security-threats-what-to-look-for>
- <https://medium.com/xord/hyperledger-fabric-architecture-a-deep-dive-cea478f39765>
- <https://espeoblockchain.com/blog/a-practical-guide-to-hyperledger-fabric-security>
- <https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0022-2>
- <https://developer.ibm.com/tutorials/cl-blockchain-private-confidential-transactions-hyperledger-fabric-zero-knowledge-proof/>
- <https://github.com/hyperledger-labs/chaincode-analyzer>