Blockchain Architecture and It's Applications (Tutorial)

Presented By

Kuheli Sai

Ph.D. Student

Department of Informatics and Networked Systems

School of Computing and Information

University of Pittsburgh

Email: kuheli.sai@pitt.edu

Presentation Date: 1st November, 2018

LABORATORY FOR EDUCATION AND RESEARCH ON SECURITY-ASSURED INFORMATION SYSTEMS

PITTSCI

Overview on Modules

- M1: Blockchain Concepts
  - -What is Blockchain?
  - -Why do we need Blockchain?
  - -Different types of Blockchain
- M2: Current researches on Blockchain
  - -Different Types of Attacks on Blockchain
- M3: Applications of Blockchain
  - -Self emerging data release using Blockchain
  - -Blockchain for Securing Healthcare Domain

Overview on Modules..(continue)

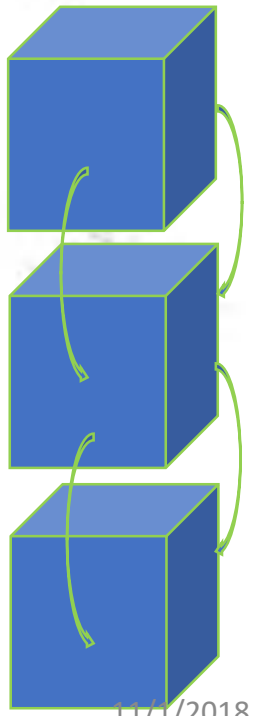
- M4: Real World Implementation of Blockchain
  -Ethereum Platform

- M5: Programming Language used to interact with Blockchain
  -Smart Contract Programming Language

- M6: Other Details
  -Brief overview on Hyperledger
  -Advantages and dis-advantages of using blockchain

Module-01

What is Blockchain?



Block

1. Computing Resources
2. Electricity

PoW

Blockchain

- It is a Public Distributed append-only Ledger, maintained by anonymous peer-to-peer nodes present in the network.
- Whenever any new transaction happens, it is listened by special types of nodes called Miners. They try to form a block by accumulating all the transactions into an abstraction of block within certain time (in case of Bitcoin it is 10 minutes).
- Miner nodes constantly tries to find a valid nonce to mine a block.
- After mining a block, miner broadcasts that block to all other peers.
- Acceptance of that block happens if the peer nodes take that block as the previous block to mine a new block.

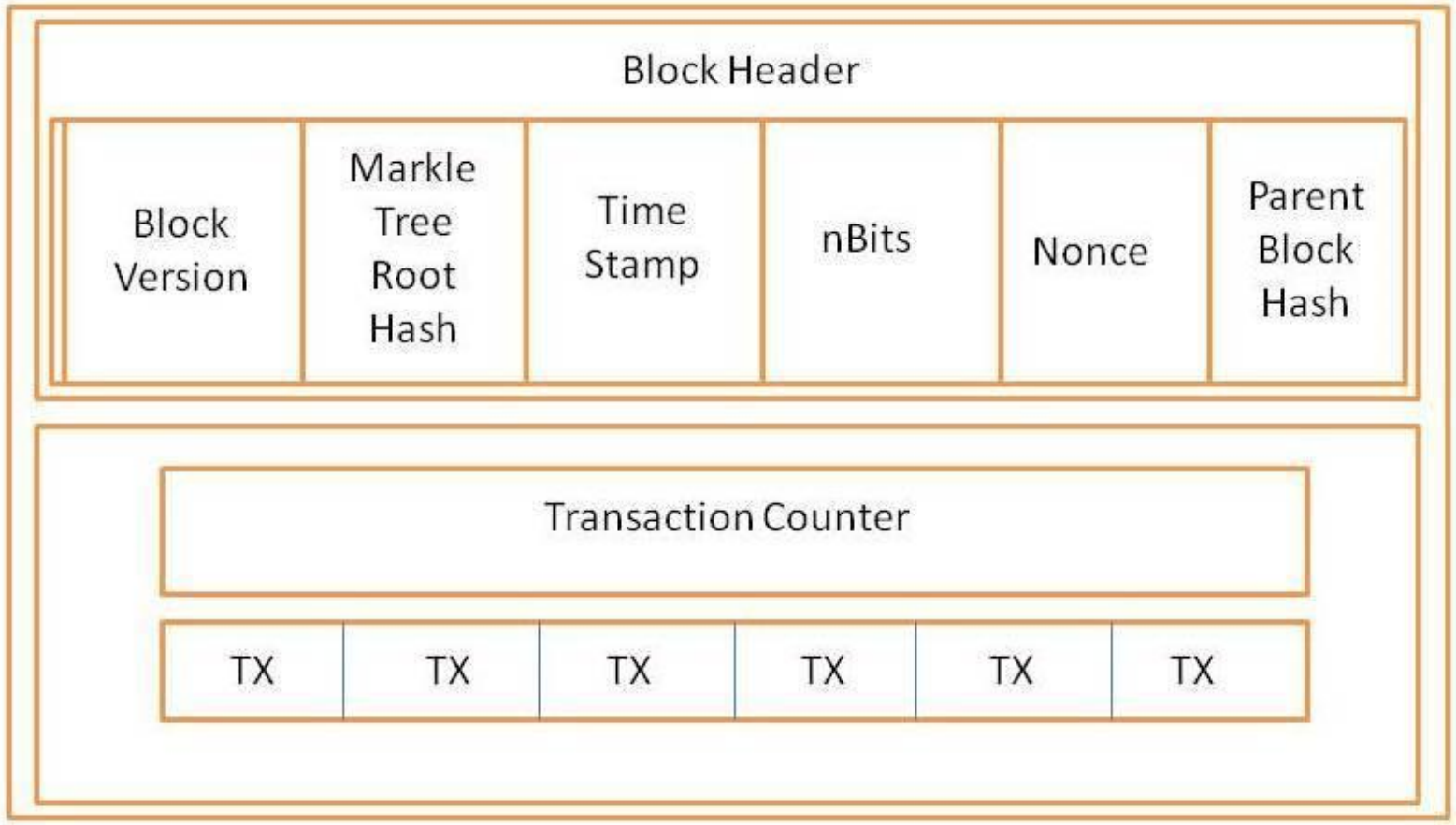
Blockchain

- Globally Shared
- Transactional Database
- Everyone can read entries of the database if they joins the network and participate there.
- If the transaction is added to the blockchain database, no one can alter that.

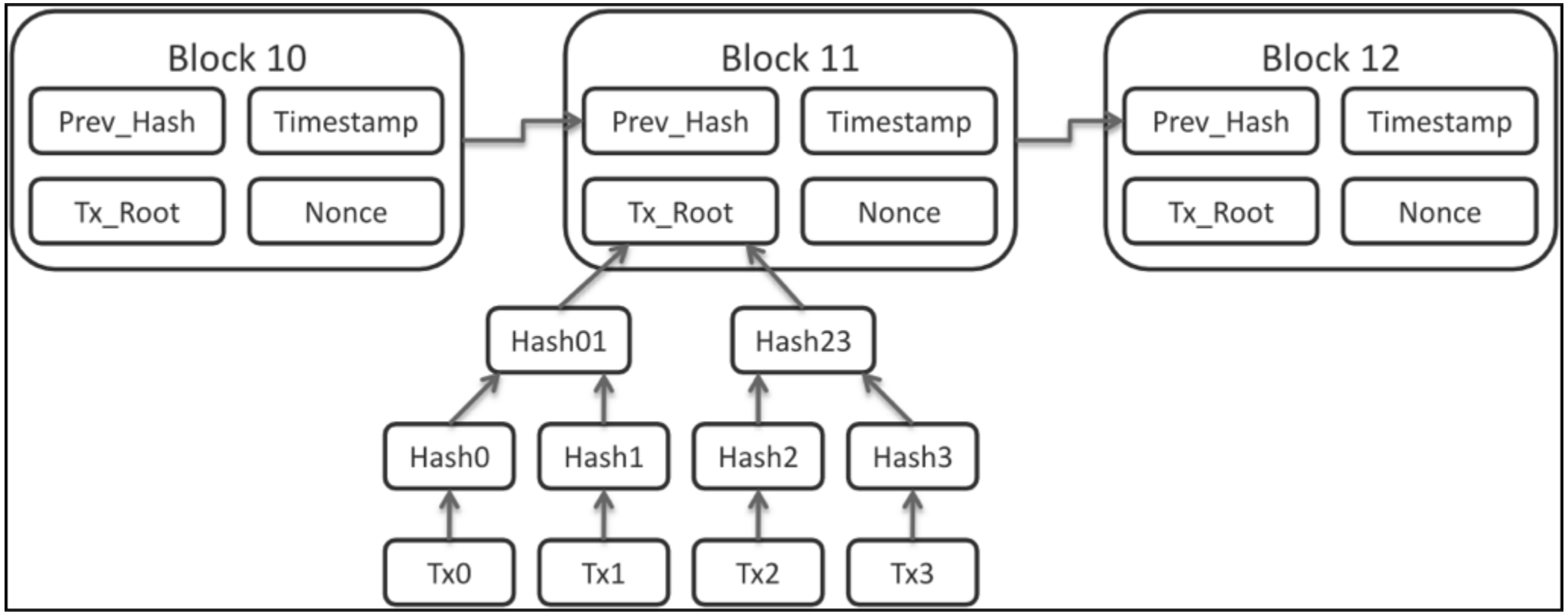
Steps to run a Blockchain Network

- New transactions are broadcasted to all the nodes
- Each node collects transactions into a block
- Each nodes works on finding a difficult proof-of-work for its block
- When a node finds a proof-of-work, it broadcasts the block to all the nodes
- Nodes accept the block only if all transactions in it are valid and not already spent
- Nodes express their acceptance of block by working on creating the next block in the chain using the hash of the accepted block as the previous hash.
- If two nodes broadcasts different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received and save the other branch in case it becomes longer.
- New transaction broadcasts do not need to reach all the nodes. As long as they reach many nodes, they will get into a block before long.
- Bitcoin like blockchain is secure from attack and consistent as long as more than half of the computational power is controlled by honest nodes.
- For mining a block and investing their resources, there exists reward based mechanism for miner nodes.

Structure of a Block in Blockchain

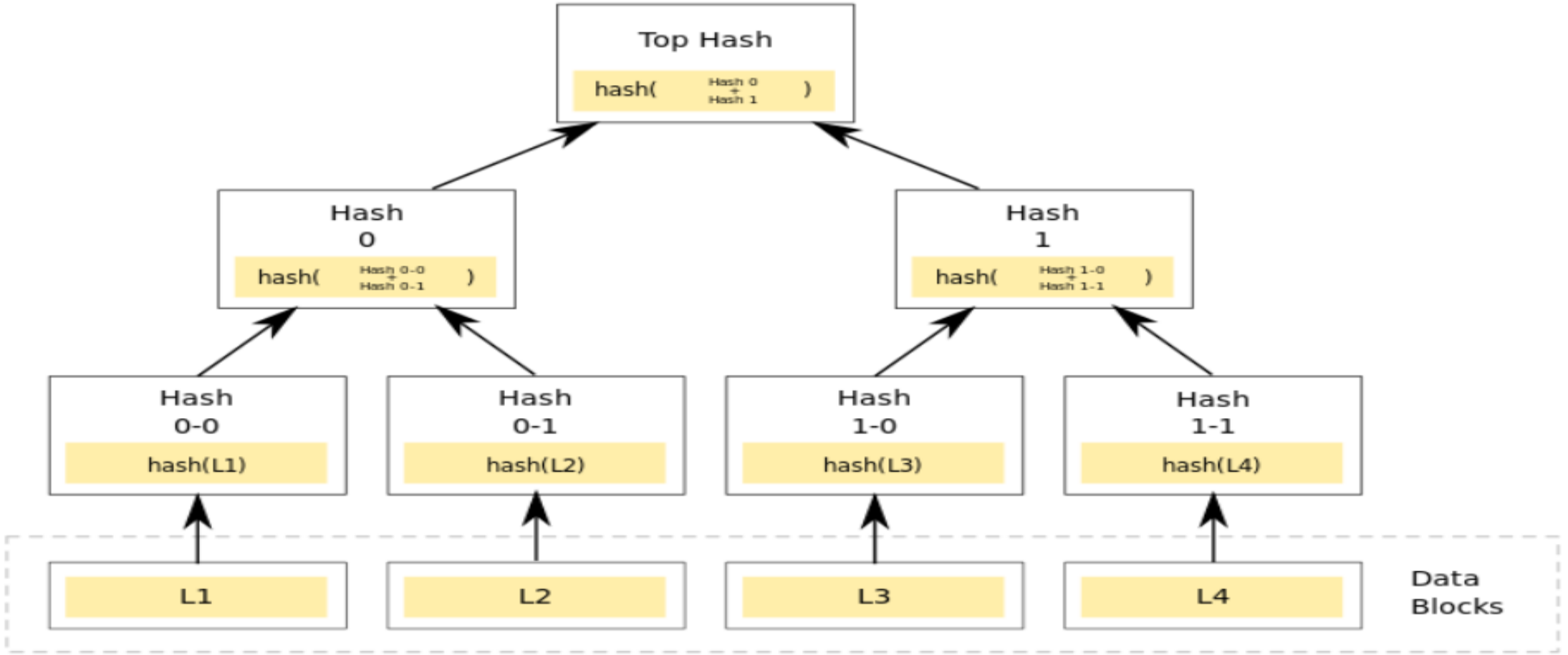


Blocks and Chain of Blocks in a Blockchain



Source: https://blockgeeks.com/guides/what-is-hashing/

Structure of a Merkle Tree



Source: https://blockgeeks.com/guides/what-is-hashing/

Why do we need Blockchain?

Why do we need Blockchain?

- Initial idea came up from the Bitcoin whitepaper where the main aim of the scientist was to demolish the conventional banking system, i.e. to create a system where we will not trust any centralized system.
- We need a mechanism where we will timestamp each transaction and everybody will accept that as the valid one.
- We need to keep the history of records that anybody cannot tamper.

Components of Blockchain

- Transaction
- Block
- Mining
- Peer to peer network
- Consensus mechanism

Different Types of Blockchain

Different Types of Blockchain

- **Public Blockchain**
- Any full node present in the network can participate in the block mining procedure
- Example: Bitcoin, Ethereum

- **Permissioned Blockchain**
- It is usually build for and maintained by a single or some sets of organizations.
- All the participants of those organizations are provided with some identity from certification authority
- So, only authorized user can participate in the Block Mining procedure
- Example: Hyperledger fabric

Different Types of Blockchain.. (continue)

- **Private Blockchain**

- It is usually used by developers and researchers to create their own private network for experimentation
- Anybody can create their own Test Network either using Ethereum TestRPC or using Ganache of Truffle

- **Hybrid Blockchain**

- It uses both public and private network
- Example: XinFin

Module-02

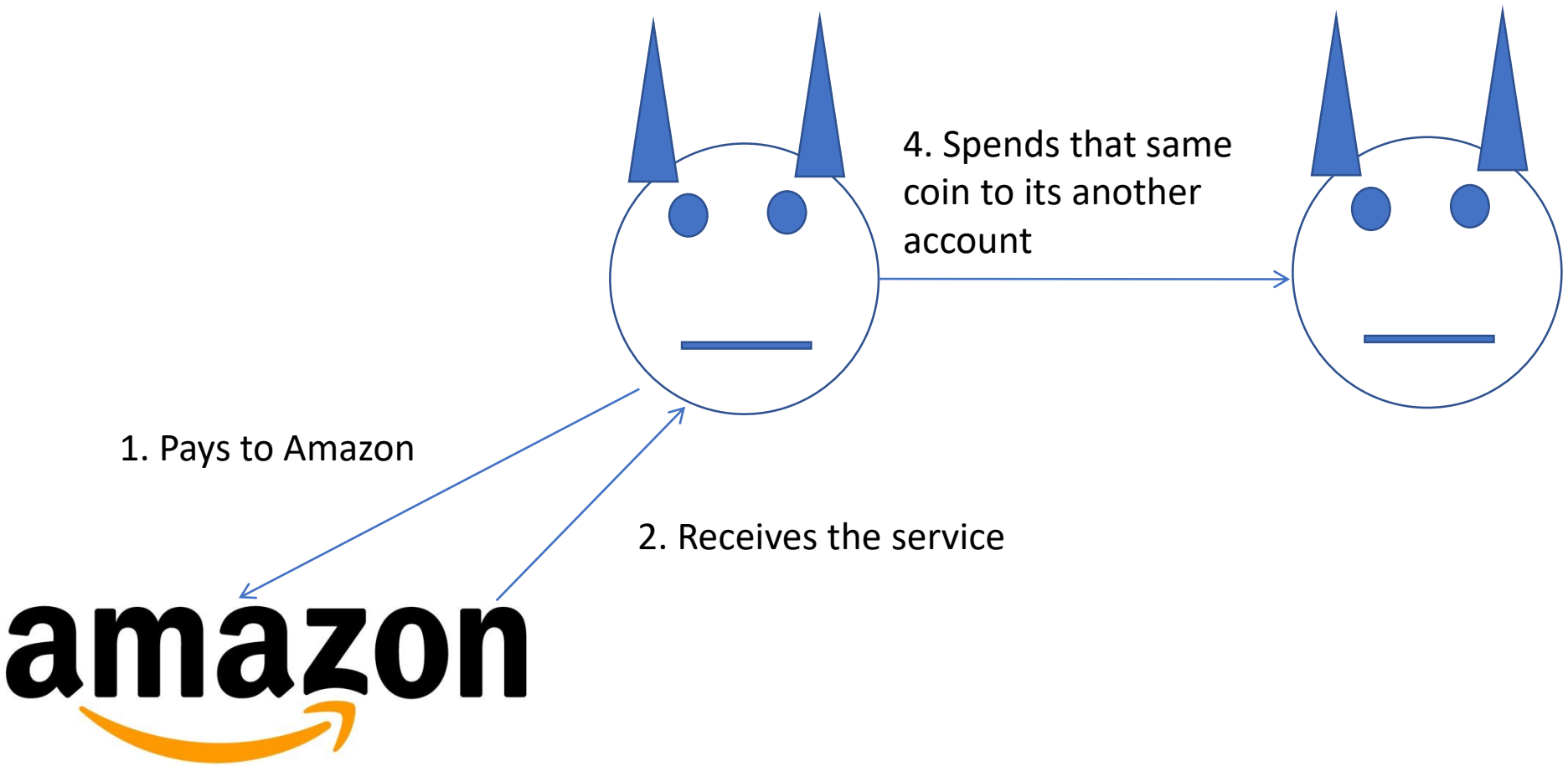
State of the art Researches on Blockchain

Current Research

- Blockchain Security
- Blockchain Systems
- Solving fundamental computer science problem leveraging Blockchain's core principles. i.e. using blockchain as the application.

Attacks on Blockchain

Attack-01: Double Spending Attack



Attack-01: Double Spending Attack

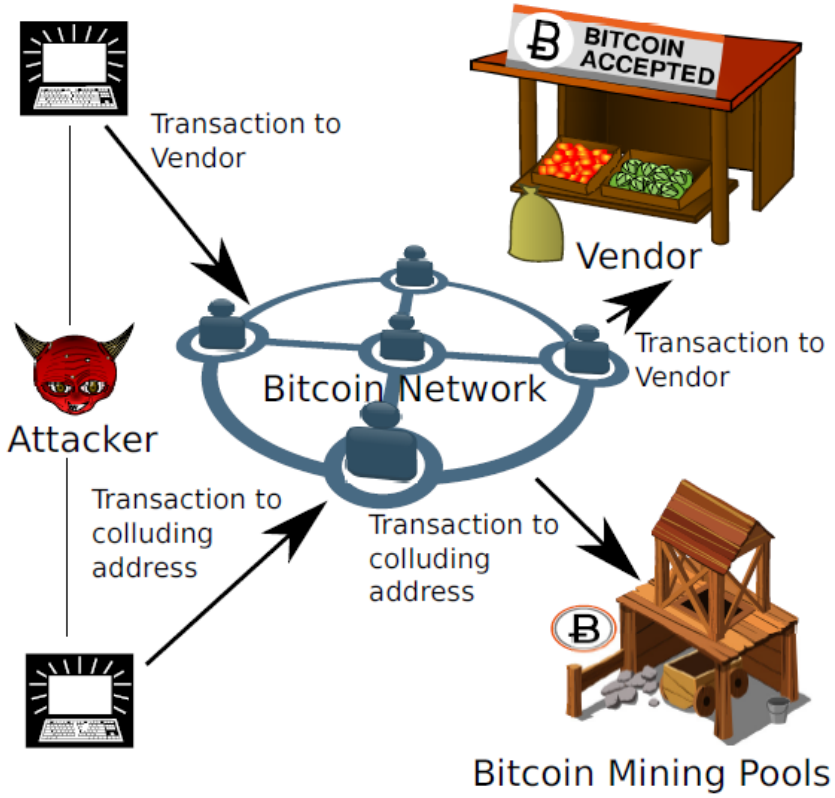
- It takes 10 minutes to mine a block
- After 6 block formation transactions in a first block gets confirmed
- Very infeasible in real life scenario where we need to get service very soon.

  Example: Paying bill in a coffee shop.

Attacks-02:
Double Spending Attack on Fast Payment Scenario

- Waiting for 10 minutes till a block is not mined is not a practical approach. Let us assume, we need to buy one coffee from a coffee store, we cannot wait these much time to get our coffee. We need some mechanism which will ensure that we will get the product on time and yet double spending won't be accomplished.

How to accomplish Double Spending in Fast Payment Scenario?



Reference Image is taken from the research paper on “Double Spending Fast Payment in Bitcoin” by Gassan O Karame

Necessary Conditions for Successful Double Spending Attacks

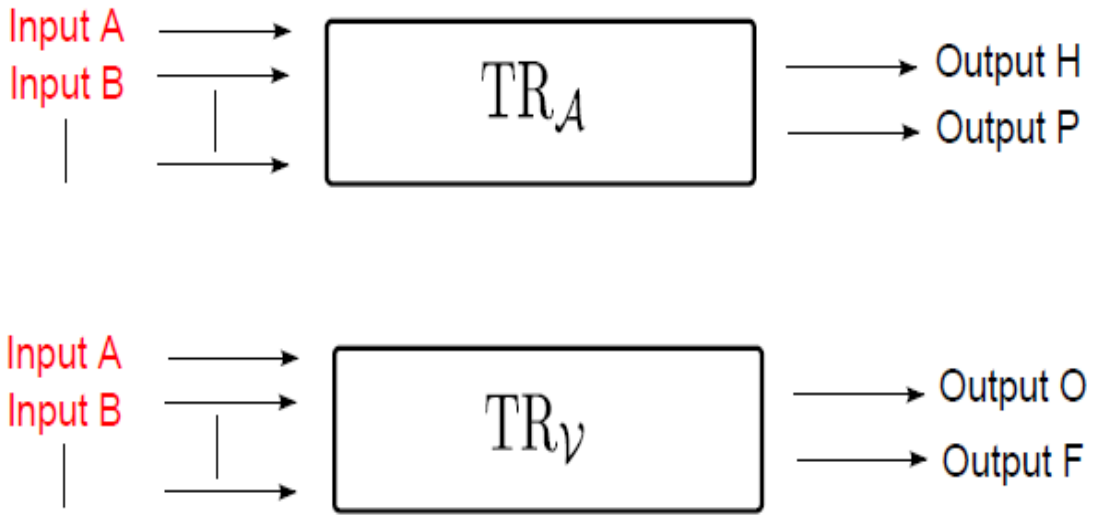


Figure: (taken from Gassan O Karame paper)
Two different transactions with the same inputs and different outputs. TR-A and TR-V can share a subset of their inputs in which A tries to double spend a subset of bitcoins that it has.

Requirement-01

TR-v is added to the wallet of V

Requirement-02

TR-A is confirmed in the wallet of blockchain

How to detect it?

- Using a listening period
- Inserting observer in the network
- Forwarding double spending attempt in the network

Attack-03:
Centralization of Hashing Power Problem

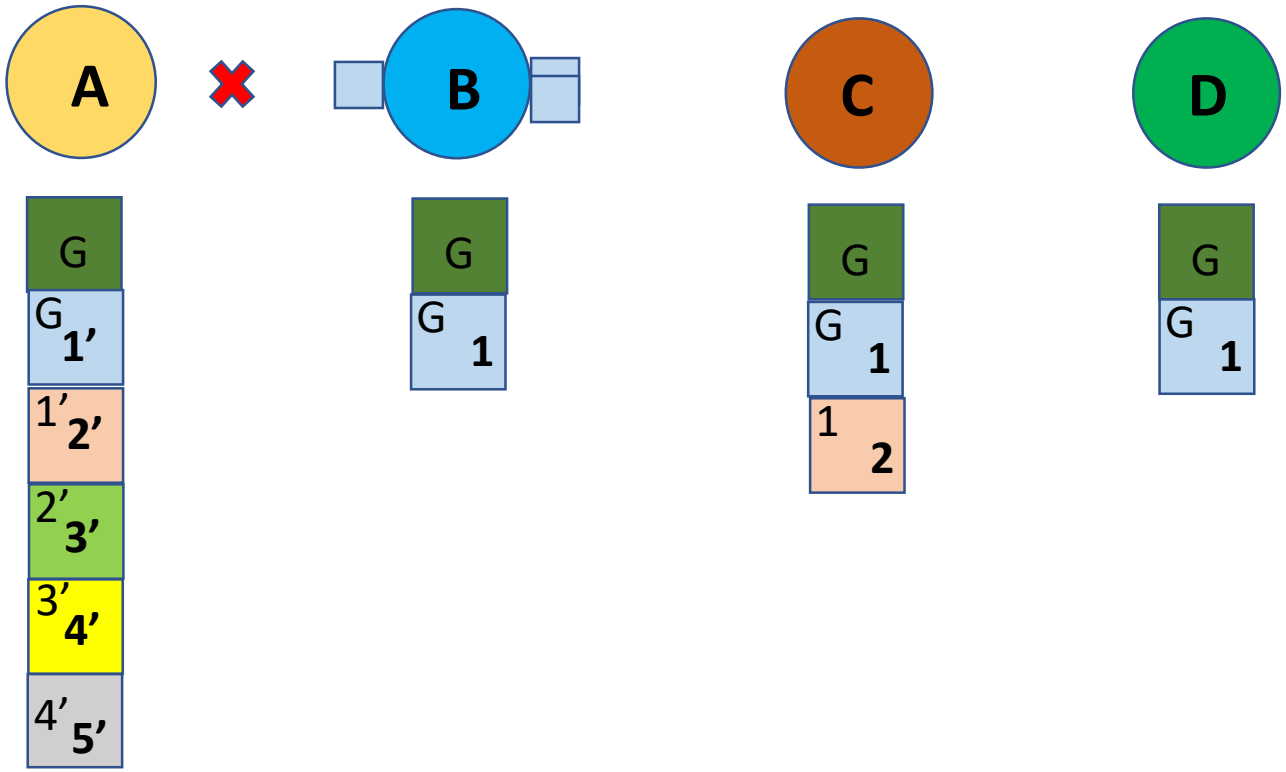
- Centralization due to mining pool
- Centralization due to server farm

- Main objective is to earn reward sooner than later.

Attack-04: Selfish Mining Attack

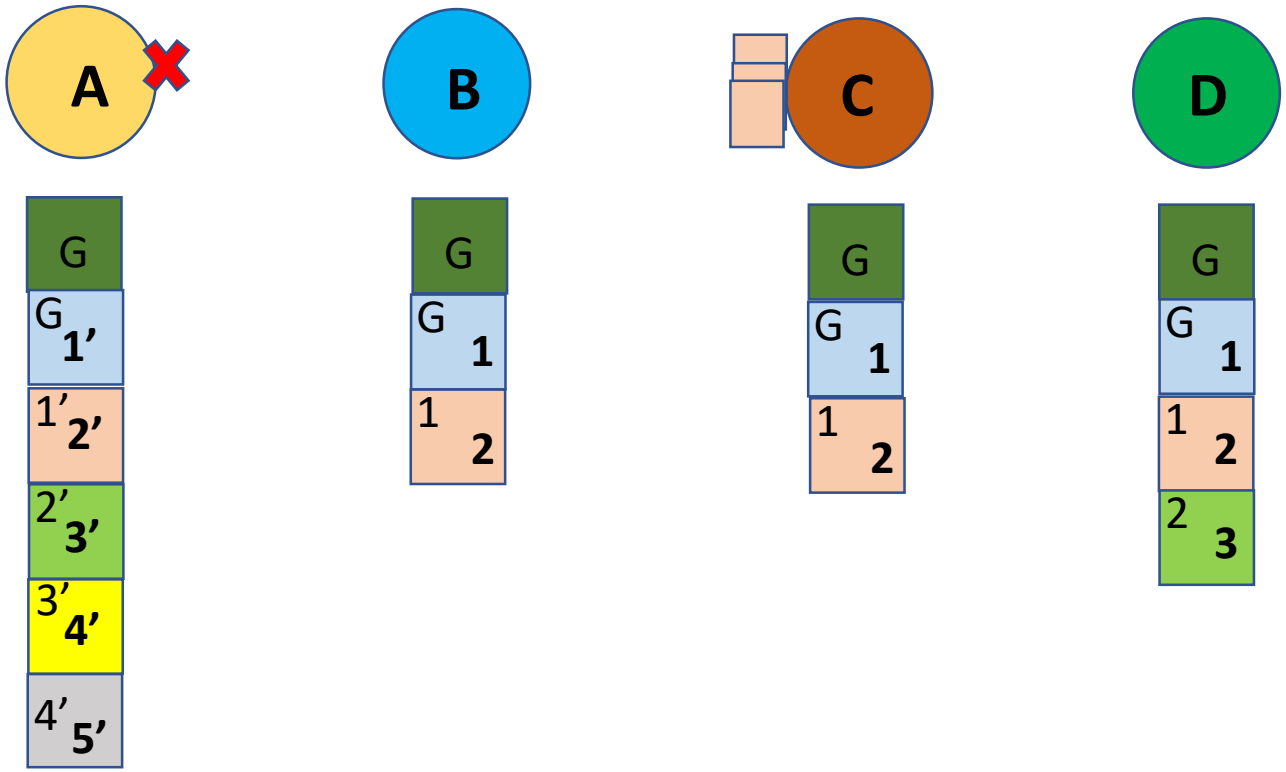
- Miner nodes form a pool.
- Objective of Pool Formation: To reduce the variation of reward obtaining time.
- Outside world the pool appears as a single node responsible for mining. Selfish miners creates block secretly and publish them only when honest miners chain approaches the secret chain.
- Objective of Selfish Mining: To invalidate honest miners effort and consequently provoking them to join colluding miners group-so that the system work as a centralized system!

Selfish Mining



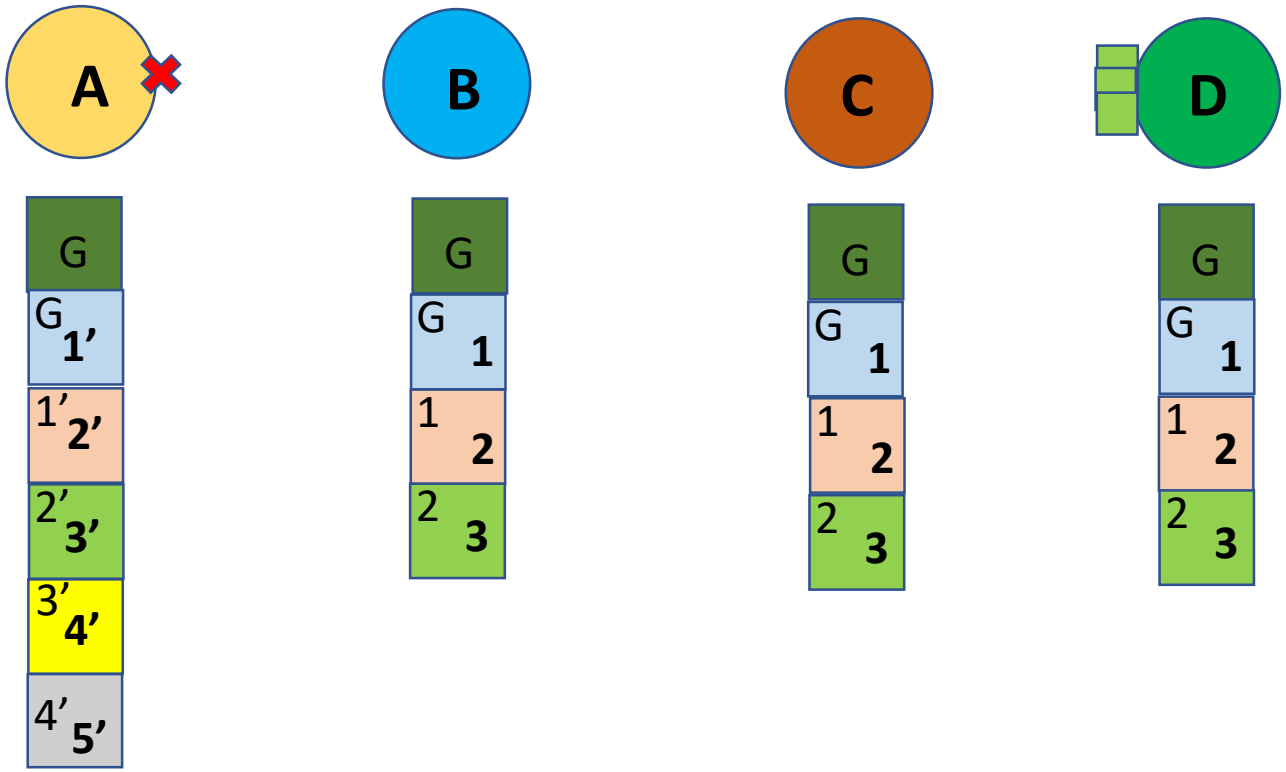
- Miner-A has mined Block-1',2',3' and 4' privately and does not publish that
- All the miners have access to their Genesis Block
- Miner –B has been chosen randomly to publish its mined block
- Miner-C and D receives the Block send by Miner-B but Miner-A discards that deliberately
- Miner-C and D validates Block-1and adds it to its own local Blockchain
- Miner-A has mined Block-5' secretly
- Miner-C has been chosen randomly to publish its mined block to other nodes

Selfish Mining



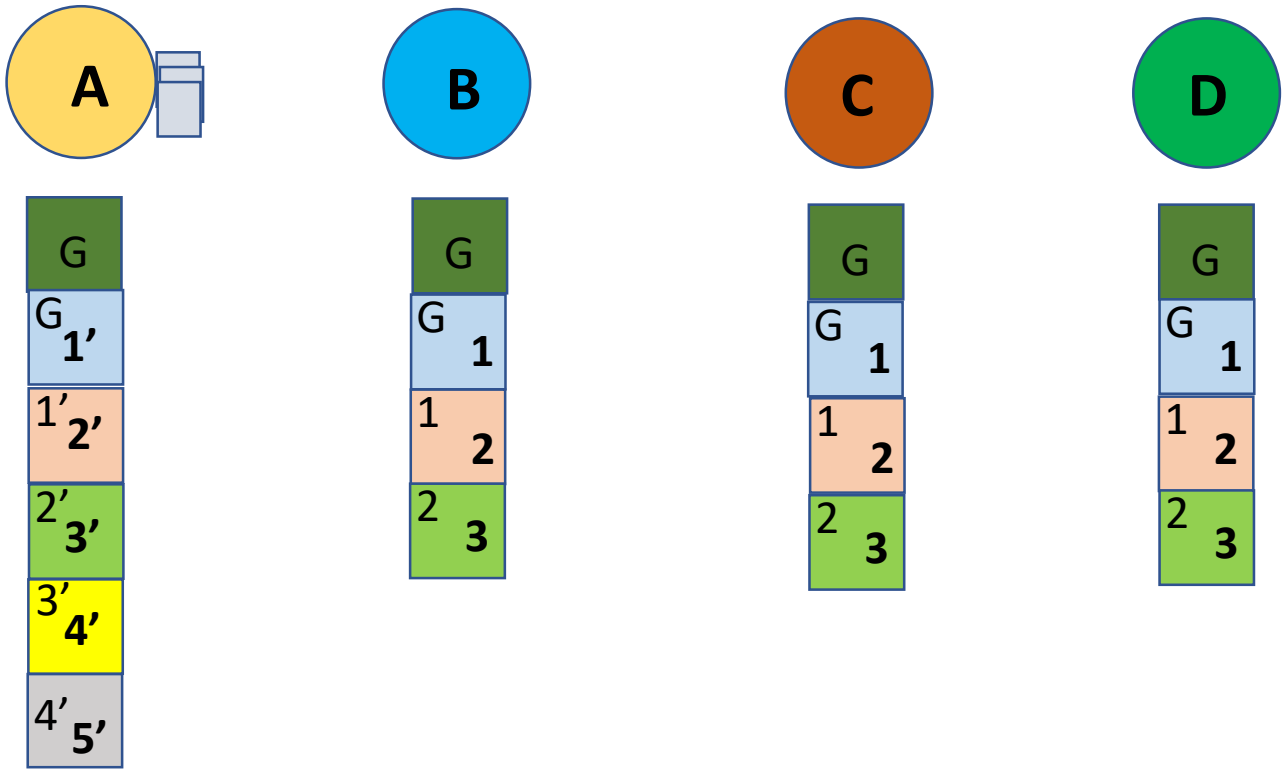
- Miner-C Broadcasts Block-2 to other miners
- Miner-B and D adds Block-2 to its own local memory and Miner-A again discards this block deliberately
- Now, Miner-D has been chosen randomly to broadcast its mined block to all other nodes

Selfish Mining



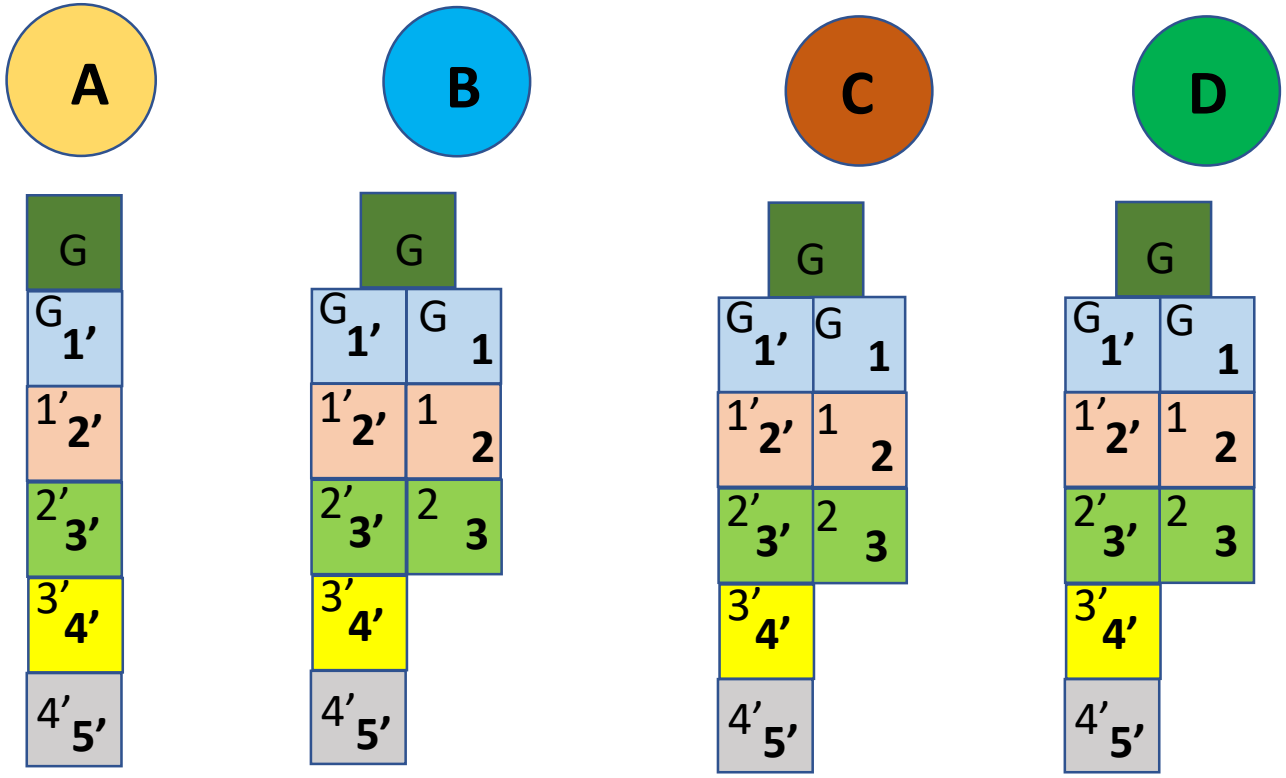
- Miner-D Broadcasts Block-3 to all the other Miners
- Miner-A discards that Block
- Miner-B and C validates that block and adds it to its own Blockchain
- Now Miner-A has been chosen randomly to broadcast its mined block to other miners

Selfish Mining



- Miner-A broadcasts its last mined block-5' to all the other miner nodes
- After getting Block-5', Miner-B finds from its previous hash that it doesn't have Block-4'. So it request Block-4' from its peers.
- After getting Block-4' from its peers, in a similar way, it finds that Block-3', Block-2', Block-1' is not present in its local memory. So it requests all the missing blocks from its peers and stores them into its local memory with the conviction that this chain might be the valid one.
- In a similar way, Miner-C and Miner-D also requests all the missing blocks from the peers and stores them into its local memory.

Selfish Mining



- Chain obtained from Miner-A is the longest one. So based on Proof-of-Work consensus mechanism, all the miners will think this chain as the valid one and will start mining in the next round taking Block-5' as the previous block.

Attacks-05: Eclipse Attack

- It is performed on the bitcoin's peer-to-peer network
- Adversary controls sufficient amount of IP addresses to monopolize all the connections to and from a victim bitcoin node
- Attacker then exploit the victim for attacks on bitcoin's mining and consensus system including N-confirmation double spending, selfish mining, and adversarial fork in the blockchain

Countermeasures

- Deterministic Eviction
- Random Selection
- Test Before Evict
- Feeler Connections
- Anchor Connections
- More Buckets
- More Outgoing Connections
- Ban unsolicited ADDR messages
- Diversify incoming connections
- Anomaly Detection

Attack-06: Block With Holding Attack

- Objective of attacker: To gain more reward or to demolish the chances of getting high-reward for all other miners
- Even after mining a valid successful block, attacker with-hold that and submit the block only when some other miner is about to publish their block.

Attack-07: Miner's Dilemma

- One kind of Block-Withholding Attack.
- Main Focus: Pool attack, i.e. each pool finds a dilemma whether or not to attack another pool.
- Result:

1. No-pool-attacks is not a Nash equilibrium. If none of the other pools attack, a pool can increase its revenue by attacking the others.

2. When two pools can attack each other, they face a version of the Prisoner's Dilemma. If one pool chooses to attack, the victim's revenue is reduced, and it can retaliate by attacking and increase its revenue.

3. However, when both attack, at Nash equilibrium both earn less than they would have if neither attacked.

4. With multiple pools of equal size a similar situation arises with a symmetric equilibrium.

Attack-07: Miner's Dilemma (continue)

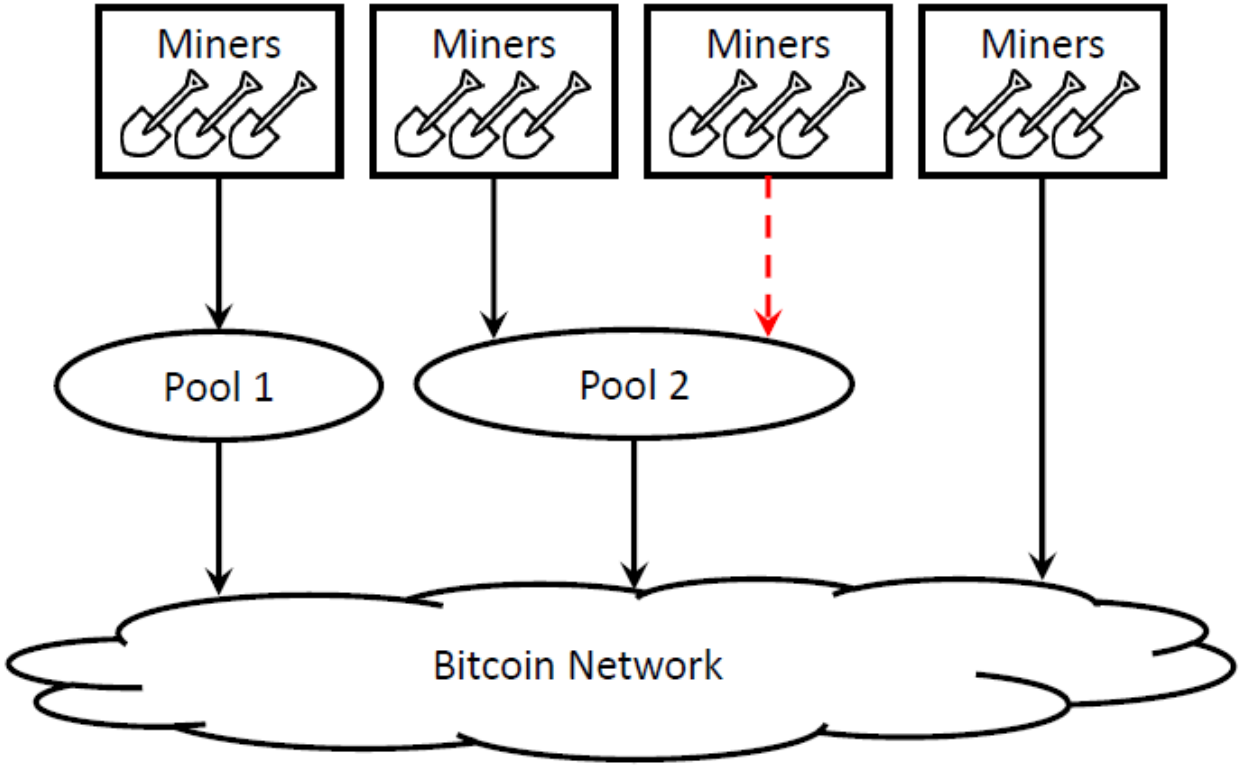


Figure:
Classical block with-holding attack. A group of miners attack Pool-02 with a block withholding attack denoted by dashed arrow red line

Source:
Miner's Dilemma research paper by Ittay Eyal

Attack-07: Miner's Dilemma (continue)

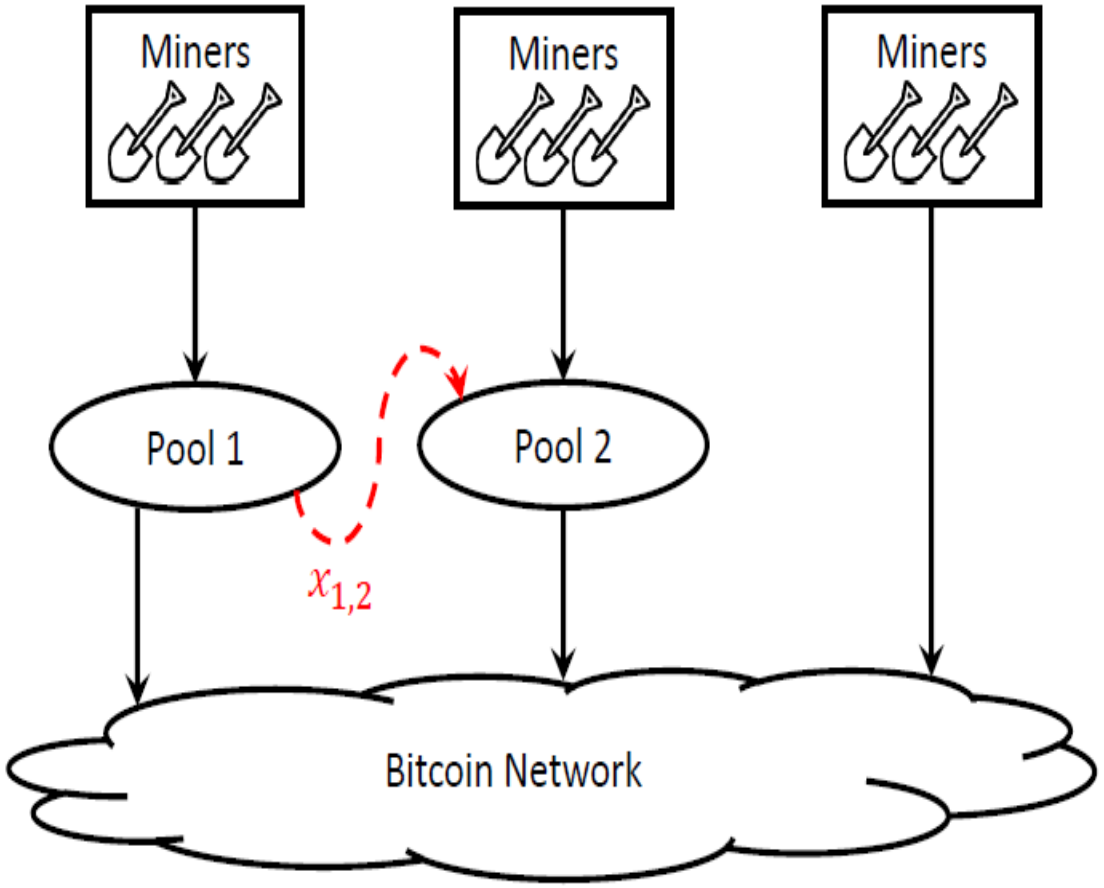


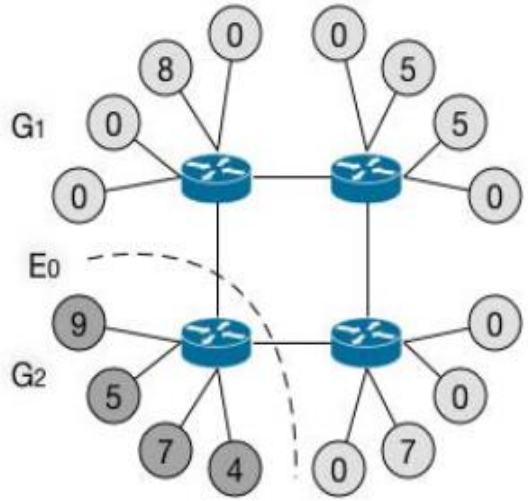
Figure:
The one attacker scenario. Pool 1 infiltrates pool 2

Source:
Miner's Dilemma research paper by Ittay Eyal

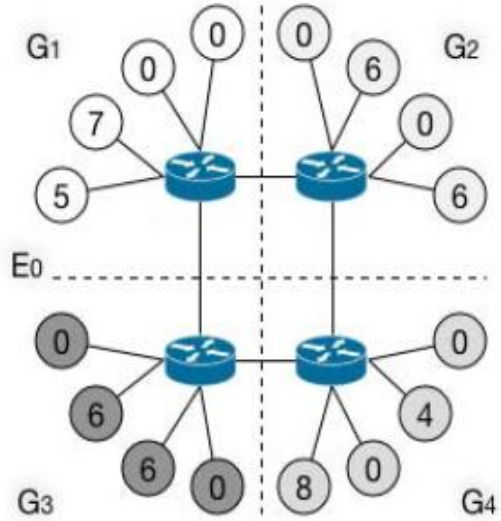
Attack-08: Balance Attack

- Most blockchains are forkable i.e. they require participants to agree on a chains out of multiple possible branches of blocks
- Main component: delaying network communications between multiple subgroups of nodes with balanced mining power.
- Single machine needs to delay messages for 20 minutes to double spend while a coalition with a third of the mining power would simply need 4 minutes to double spend with 94% success.

Balance Attack



(a) Example of the selection of edges E_0 delayed between $k = 2$ subgraphs of 25 units of mining power each



(b) Example of the selection of edges E_0 delayed between $k = 4$ subgraphs of 12 units of mining power each

Reference image is taken from the research paper on the Balance Attack by Christopher Natoli

Module-03

Applications of Blockchain

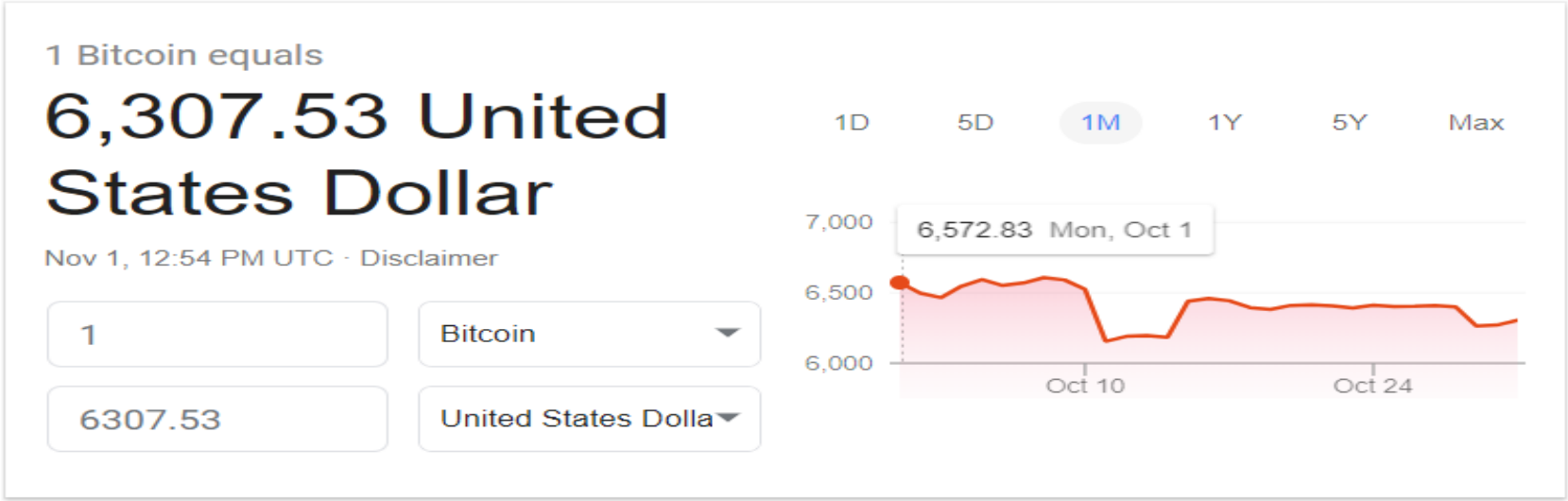
Applications of Blockchain

- Cryptocurrency
- Supply Chain Management
- Food Sector (Example: Walmart)
- Know Your Customer (KYC)
- Diamond Tracking
- Time Release of Self Emerging Data
- Health-Care System

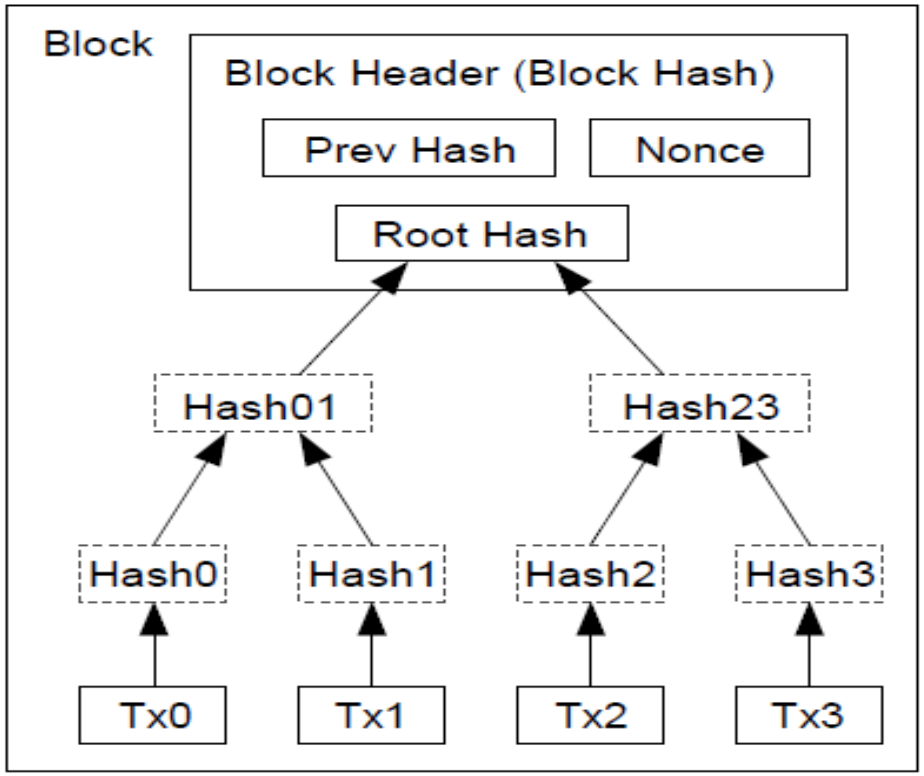
Application-01: Cryptocurrency

Bitcoin

- First the concepts came from the Bitcoin whitepaper of Satoshi Nakamoto
- Time: During 2008
- Main objective: Time stamp transaction without using centralized authority
- Disadvantage: Huge energy consumption for mining a block
- As of today (1st November, 2018), 1 BTC= 6307.53 USD



Bitcoin

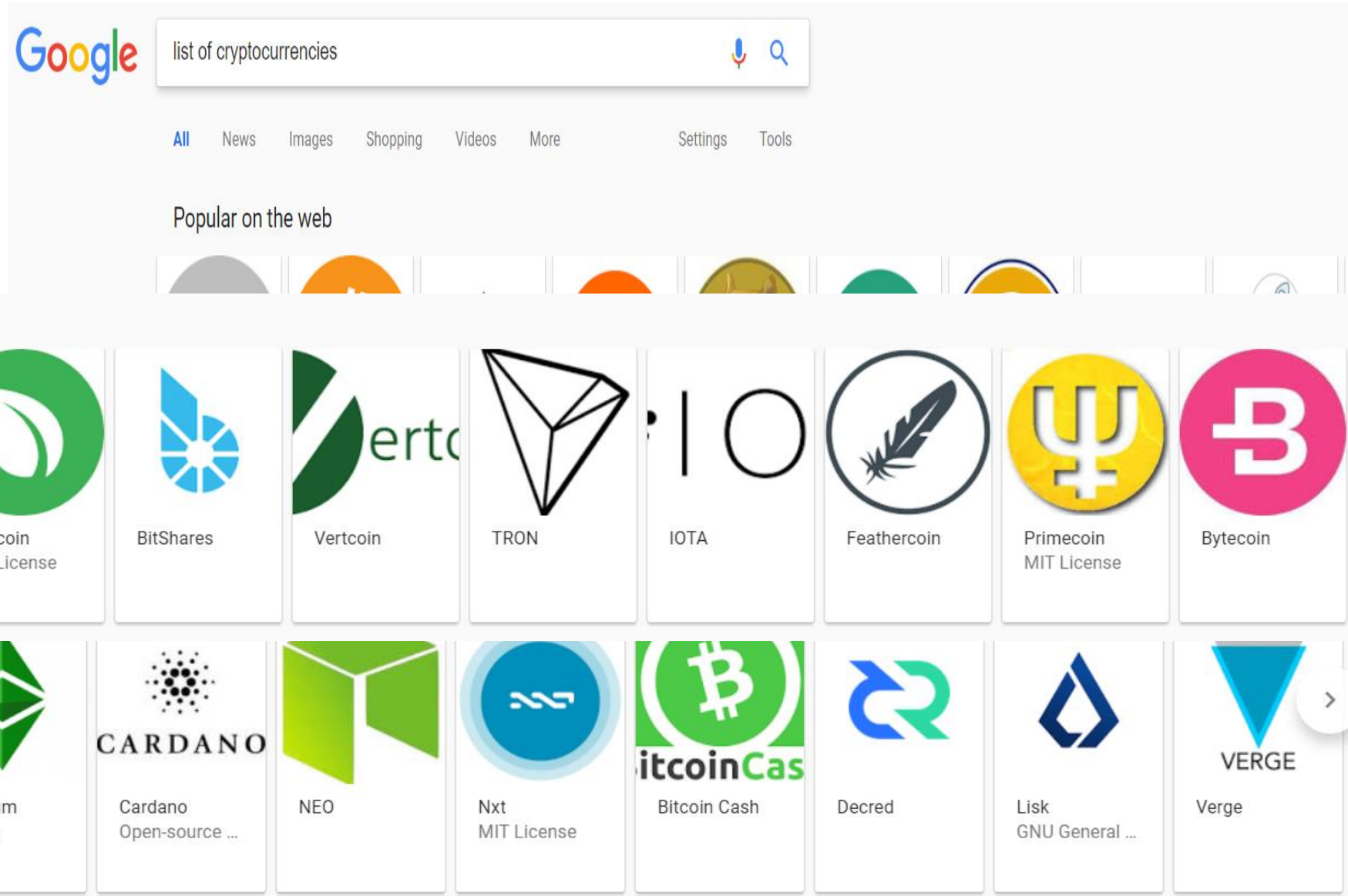


Transactions Hashed in a Merkle Tree

Source: Bitcoin whitepaper

Other Cryptocurrencies

- Bitcoin
- Litecoin
- Ether of Ethereum
- Dogecoin

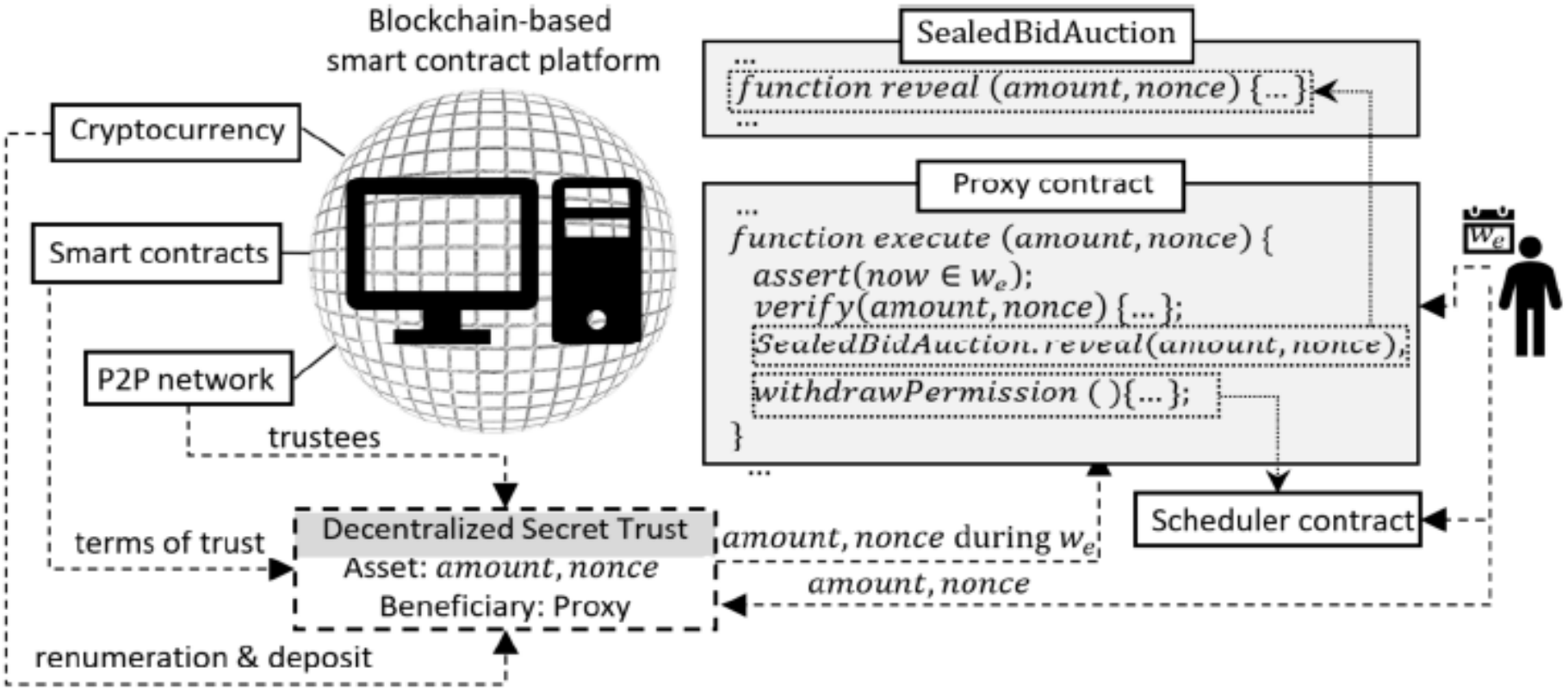


Application-02:
Time Release of Self Emerging Data using Blockchain

Time Release of Self Emerging Data

- When the data will be released in future point of time will be measured and done by using blockchain
- Example of future release of data: online examination question paper that should be available only after a certain time, not before that. In fact it has to be available fully from the time it should be available.

Figure:
At time t1, Bob wants to schedule function reveal(amount,nonce) in contract SealedBidAuction [17] to be executed during a future time window We



Source: Research paper on “Decentralized Privacy-preserving Timed Execution in Blockchain-based Smart Contract Platforms” by Chao Li

Possible Attacks

- The Difference Attack
  - Malicious Trustee
  - Trustee Identity Disclosure
  - Advance Disclosure

- Execution Failure Attack
  - Absent Trustee
  - Fake Submission

Protocol

- To overcome all the mentioned problem, a protocol is designed using Ethereum smart-contract feature which has several components which are-
  1. Trustee Application
  2. User Scheduler
  3. Misbehavior report

Outcome

- This new mechanism allows users of Ethereum-based decentralized applications to schedule timed transactions without revealing sensitive inputs before an execution time window chosen by the users
- No centralized party is needed here
- Allows users to go offline at their discretion after scheduling a timed transaction.
- Timed execution mechanism protects the sensitive inputs by employing a set of trustees from the decentralized blockchain network to enable the inputs to be revealed only during the execution time.
- Proposed approach has been implemented using Solidity and evaluated the system on the Ethereum official test network.
- Low gas cost and low time overhead is associated with the proposed approach.

Application-03:
Blockchain in Healthcare Security

Healthcare Systems

- What is meant by Healthcare System?
- Types of attacks on the healthcare system
- Why healthcare is prone to cyber-attacks?
- How to preserve security and privacy of healthcare system?

Health-Care System

- What is Healthcare Systems?
- What is Health Data?
- What are the type/sources of health data?
- Why it is Important?
- Who cares if we loss our data?

1. Patient-names
2. Dates of Birth
3. Medical record numbers
4. Social security number
5. Insurance information
6. Provider's name
7. Dates of service
8. Health insurer
9. Medical conditions
10. Locations
11. Driver license number
12. Financial Information

Why healthcare system is prone to attack?

- Attacker can **sale** it
    1. $158 for non-health related data, $355 for health-related data
    2. $2 for PII and $363 for PHI
- Attacker can **gain personal benefit** from it
    1. Illegal access to prescriptions
    2. Can take advantage of patient's medical condition
    3. Create fake medical insurance
- Sometimes, though very less, they **tries to help** a few people by attacking the system

List of Attacks on Healthcare System and some privacy concerns of patients data

- Ransomware Attack

[Ex: Hollywood Presbyterian Hospital at California, infected due to outdated JBoss server, paid $17,000 to re-gain access to files and their network]

- DDoS Attack

[Ex: Boston Children's Hospital in 2014, spent $300,000 to mitigate the damage]

- Insider Threat

[Ex: One hospital of Texas]

- Business email compromise and fraud scams

[On 2015, a local hospital reported about a false phone call from a pharmacy to confirm $500,000 worth prescription drugs]

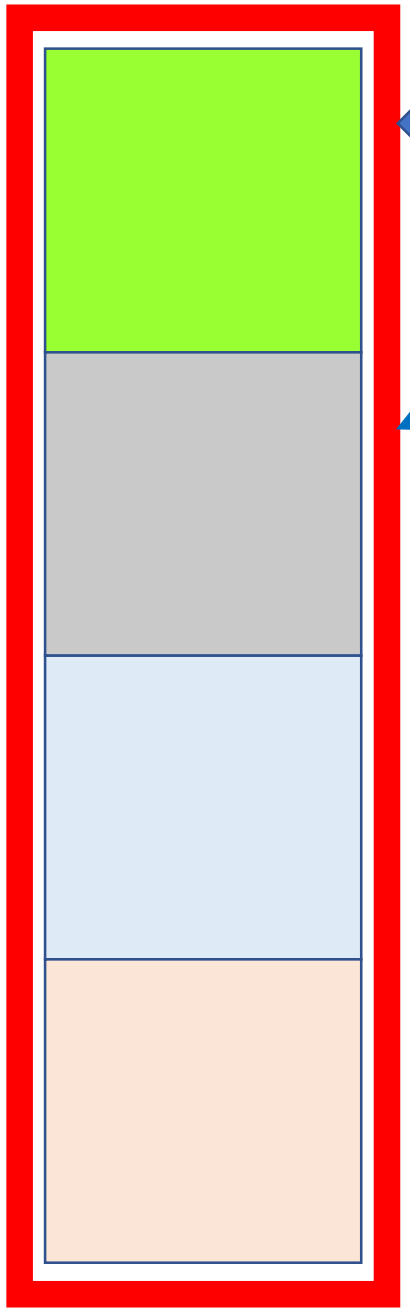
- Patient don't have any control on how their data is going to use

Some facts related to Health-Care Attacks

Company	Patient Record Compromised	Date	How they find	What Information were taken	Recovery Measure
Henry Ford Health System	18,470	Oct 3, 2017	email credentials of a group of employees were stolen	Patients records	1. multi-factor authentication 2. Issuing new medical no to patients
Arkansan Provider	1,28,000 [Ransomeware]	July 25, 2017			All impacted patients are being offered a year of free credit monitoring
AU Medical Center	1% patient data	April 20, 2017			Upon discovering the breach, officials said access to email accounts was disabled and passwords were reset.
1. Emony Brain health centre 2. Bron Lebanon hospital	26000 MongoDB Database servers [Ransomware]	Late 2016	Found by security researcher Victor Gevers	2,00,000 patients records	

Need a Robust Solution!!!!!

But HOW?

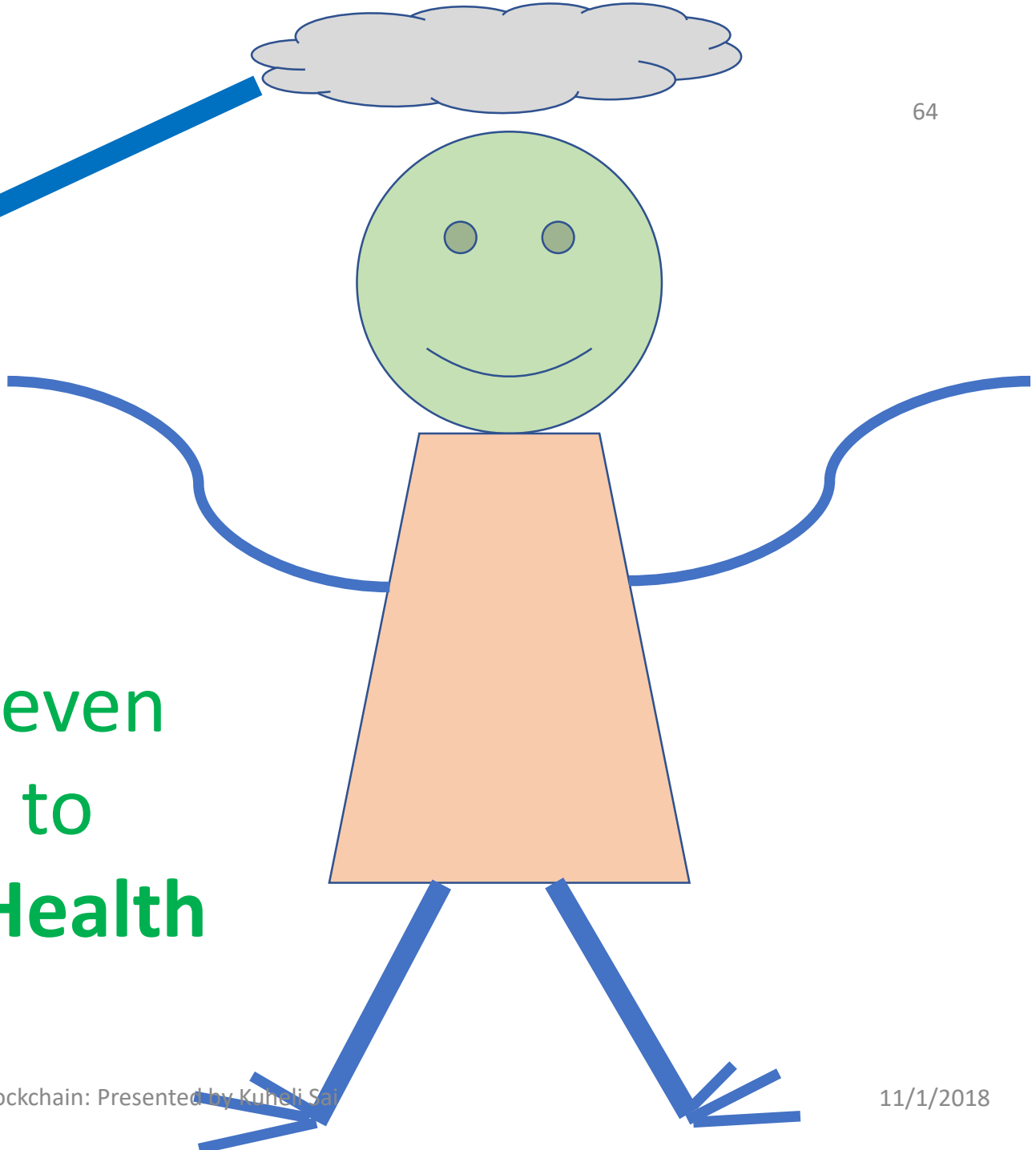


Yayyyy!!!!

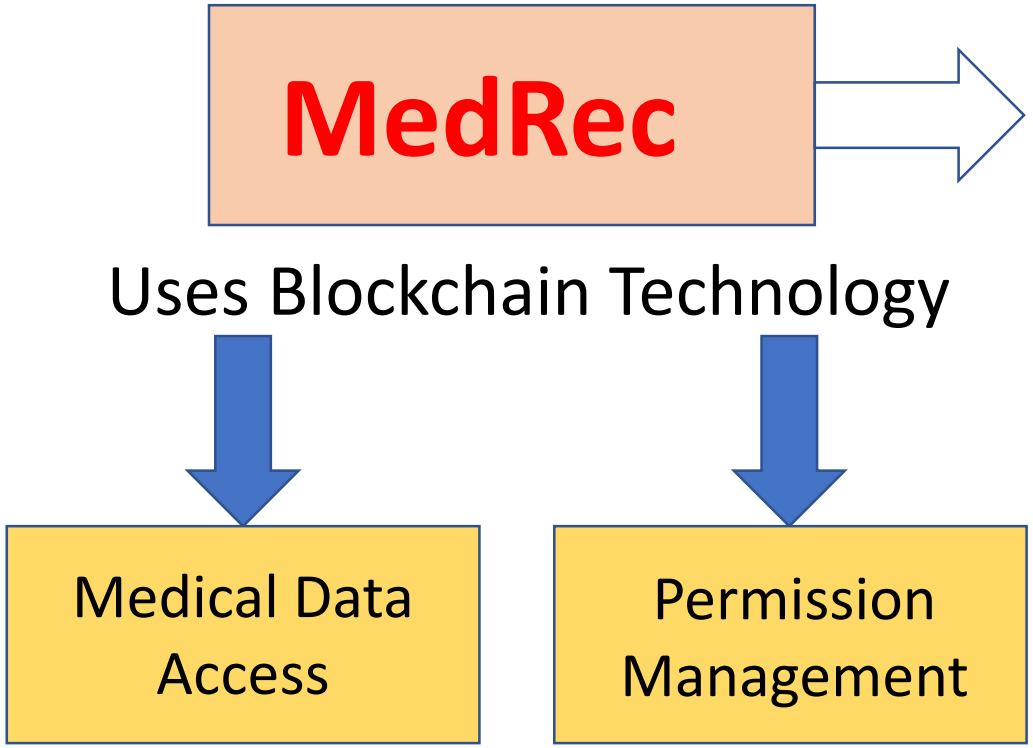
We have

Blockchain

May be we even can use it to protect our **Health Data**

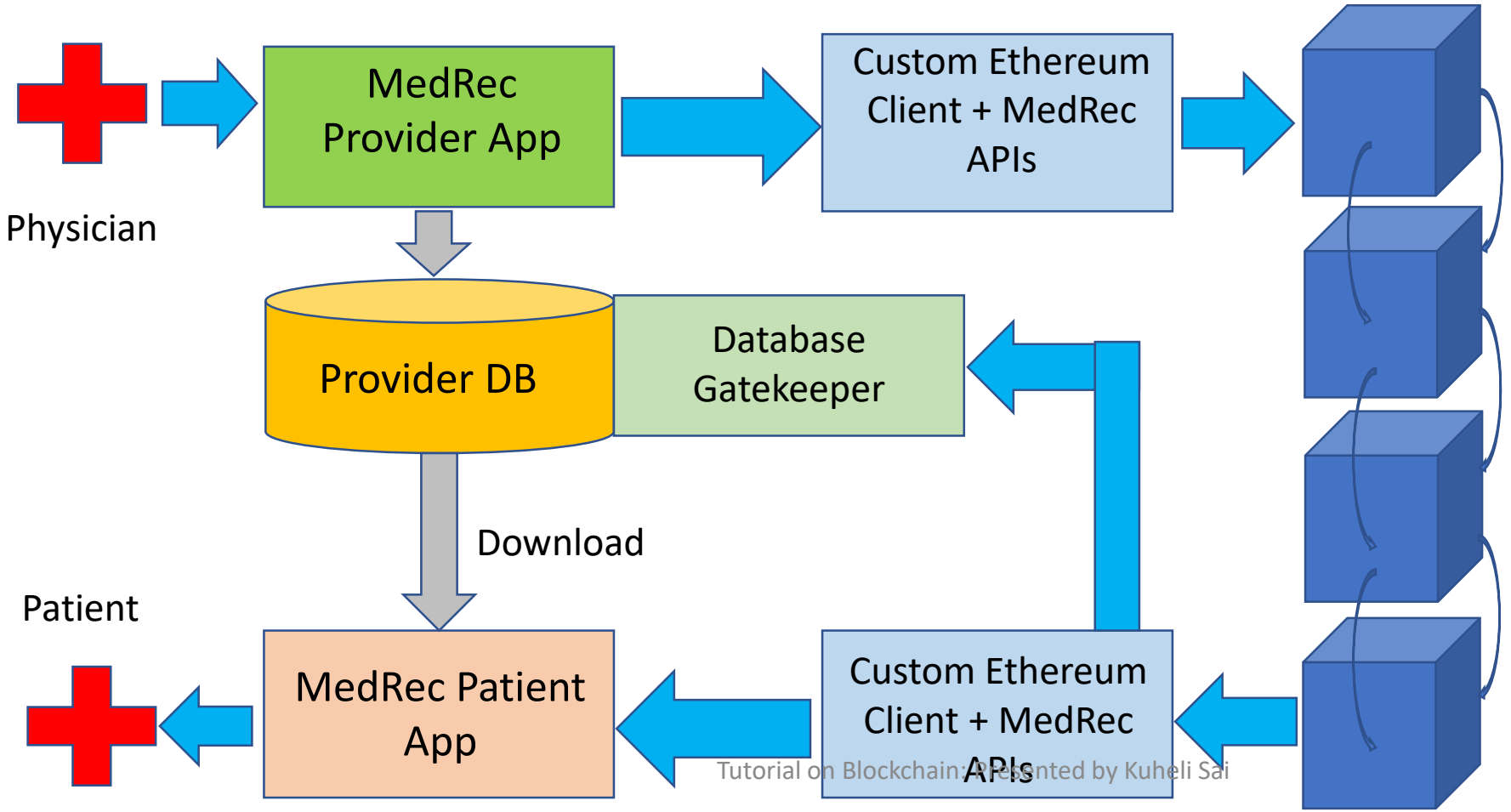


Solution Approach



Decentralized Record Management System

Solution Approach



- Private Blockchain, smart-contract
- Governs access and permissions
- Mined by researchers

Module-04

Ethereum Platform

Ethereum Platform

- What is Ethereum?
  - Decentralized platform that runs smart-contract.
  - Turing complete in nature. We can deploy any types of applications.
- What is Ethereum Virtual Machine (EVM)?
  - Runtime environment for smart-contract applications.
  - Code running inside EVM, does not have access to other code, file-system or network.
  - One smart-contract have limited access to other smart-contract.
- Why do we need it?
  - For deploying our smart-contract applications
- What is it's applications?

  -We can create any types of application we use for, be it voting using smart contract, blind auction, payment channel etc.

Components of Ethereum Virtual Machine

- Accounts
- Transactions
- Gas
- Storage, memory and stack
- Instruction set
- Message Calls
- Delegate Call/ Call code and Libraries
- Logs
- Create
- Self destruct

Module-05

Solidity:
Smart Contract Programming Language

Solidity: Smart Contract Programming Language

- Solidity is a smart contract programming language.
- It is used for taking advantage of Ethereum blockchain and deploy application on the blockchain.
- Similar to JavaScript programming language.
- There are many versions of it. Till todays date, it has total 25 versions and the language is in its inception stage currently.

Example Smart Contract Applications

Sample Application Program

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

Source: Solidity website, URL: https://solidity.readthedocs.io/en/v0.4.24/index.html

Application-01: Voting

```
pragma solidity ^0.4.22;

/// @title Voting with delegation.
contract Ballot {
    // This declares a new complex type which will
    // be used for variables later.
    // It will represent a single voter.
    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    // This is a type for a single proposal.
    struct Proposal {
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    // This declares a state variable that
    // stores a `Voter` struct for each possible address.
    mapping(address => Voter) public voters;

    // A dynamically-sized array of `Proposal` structs.
    Proposal[] public proposals;
```

Source: Solidity website, URL: https://solidity.readthedocs.io/en/v0.4.24/index.html

```
    /// Delegate your vote to the voter `to`.
    function delegate(address to) public {
        // assigns reference
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "You already voted.");

        require(to != msg.sender, "Self-delegation is disallowed.");

        // Forward the delegation as long as
        // `to` also delegated.
        // In general, such loops are very dangerous,
        // because if they run too long, they might
        // need more gas than is available in a block.
        // In this case, the delegation will not be executed,
        // but in other situations, such loops might
        // cause a contract to get "stuck" completely.
        while (voters[to].delegate != address(0)) {
            to = voters[to].delegate;

            // We found a loop in the delegation, not allowed.
            require(to != msg.sender, "Found loop in delegation.");
        }

        // Since `sender` is a reference, this
        // modifies `voters[msg.sender].voted`
        sender.voted = true;
        sender.delegate = to;
        Voter storage delegate_ = voters[to];
        if (delegate_.voted) {
            // If the delegate already voted,
            // directly add to the number of votes
            proposals[delegate_.vote].voteCount += sender.weight;
        } else {
            // If the delegate did not vote yet,
            // add to her weight.
            delegate_.weight += sender.weight;
        }
    }
```

```
    /// Give your vote (including votes delegated to you)
    /// to proposal `proposals[proposal].name`.
    function vote(uint proposal) public {
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "Already voted.");
        sender.voted = true;
        sender.vote = proposal;

        // If `proposal` is out of the range of the array,
        // this will throw automatically and revert all
        // changes.
        proposals[proposal].voteCount += sender.weight;
    }

    /// @dev Computes the winning proposal taking all
    /// previous votes into account.
    function winningProposal() public view
            returns (uint winningProposal_)
    {
        uint winningVoteCount = 0;
        for (uint p = 0; p < proposals.length; p++) {
            if (proposals[p].voteCount > winningVoteCount) {
                winningVoteCount = proposals[p].voteCount;
                winningProposal_ = p;
            }
        }
    }

    // Calls winningProposal() function to get the index
    // of the winner contained in the proposals array and then
    // returns the name of the winner
    function winnerName() public view
            returns (bytes32 winnerName_)
    {
        winnerName_ = proposals[winningProposal()].name;
    }
}
```

Application-02: Blind Auction

```
    /// End the auction and send the highest bid
    /// to the beneficiary.
    function auctionEnd() public {
        // It is a good guideline to structure functions that interact
        // with other contracts (i.e. they call functions or send Ether)
        // into three phases:
        // 1. checking conditions
        // 2. performing actions (potentially changing conditions)
        // 3. interacting with other contracts
        // If these phases are mixed up, the other contract could call
        // back into the current contract and modify the state or cause
        // effects (ether payout) to be performed multiple times.
        // If functions called internally include interaction with external
        // contracts, they also have to be considered interaction with
        // external contracts.

        // 1. Conditions
        require(now >= auctionEnd, "Auction not yet ended.");
        require(!ended, "auctionEnd has already been called.");

        // 2. Effects
        ended = true;
        emit AuctionEnded(highestBidder, highestBid);

        // 3. Interaction
        beneficiary.transfer(highestBid);
    }
}
```


Source: Solidity website

Application-03: Safe Remote Purchase

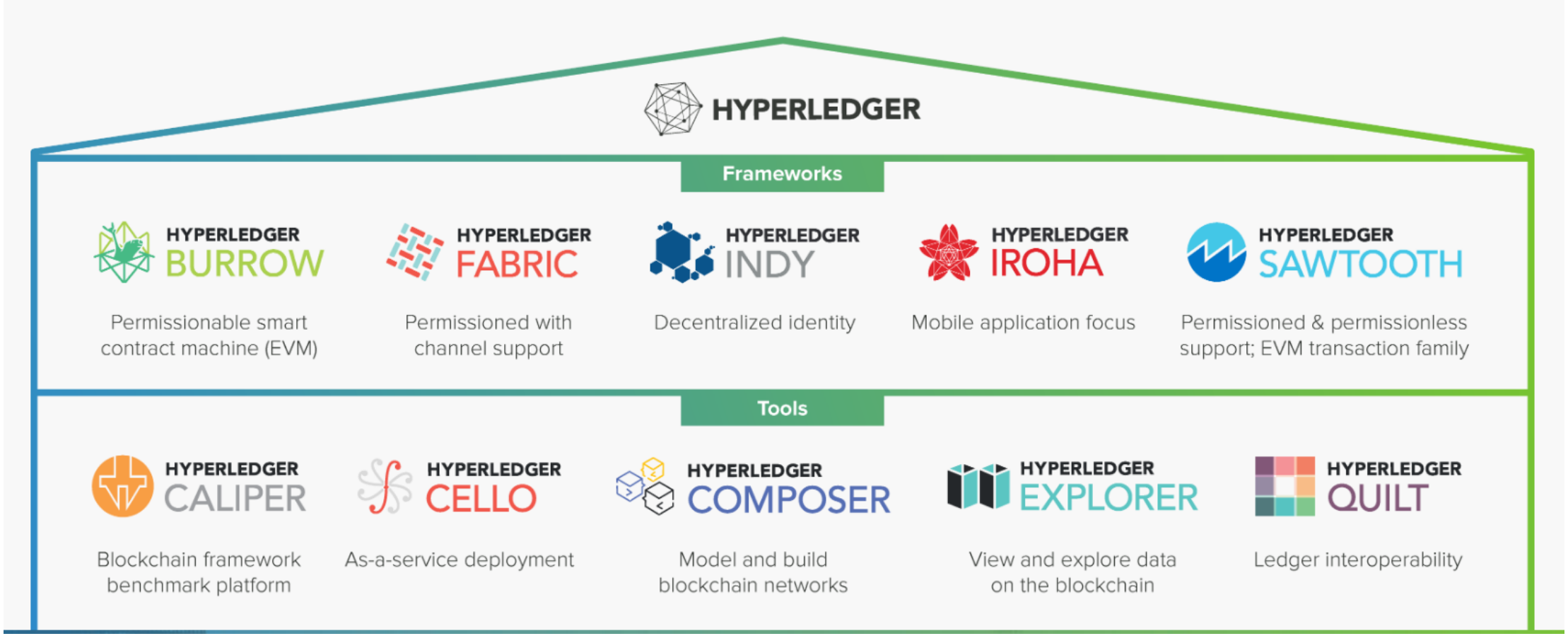
```
    /// Confirm that you (the buyer) received the item.
    /// This will release the locked ether.
    function confirmReceived()
        public
        onlyBuyer
        inState(State.Locked)
    {
        emit ItemReceived();
        // It is important to change the state first because
        // otherwise, the contracts called using `send` below
        // can call in again here.
        state = State.Inactive;

        // NOTE: This actually allows both the buyer and the seller to
        // block the refund - the withdraw pattern should be used.

        buyer.transfer(value);
        seller.transfer(address(this).balance);
    }
}
```


Source: Solidity website

Module-06

Overview on Hyperledger

Hyperledger



Source: Hyperledger Website

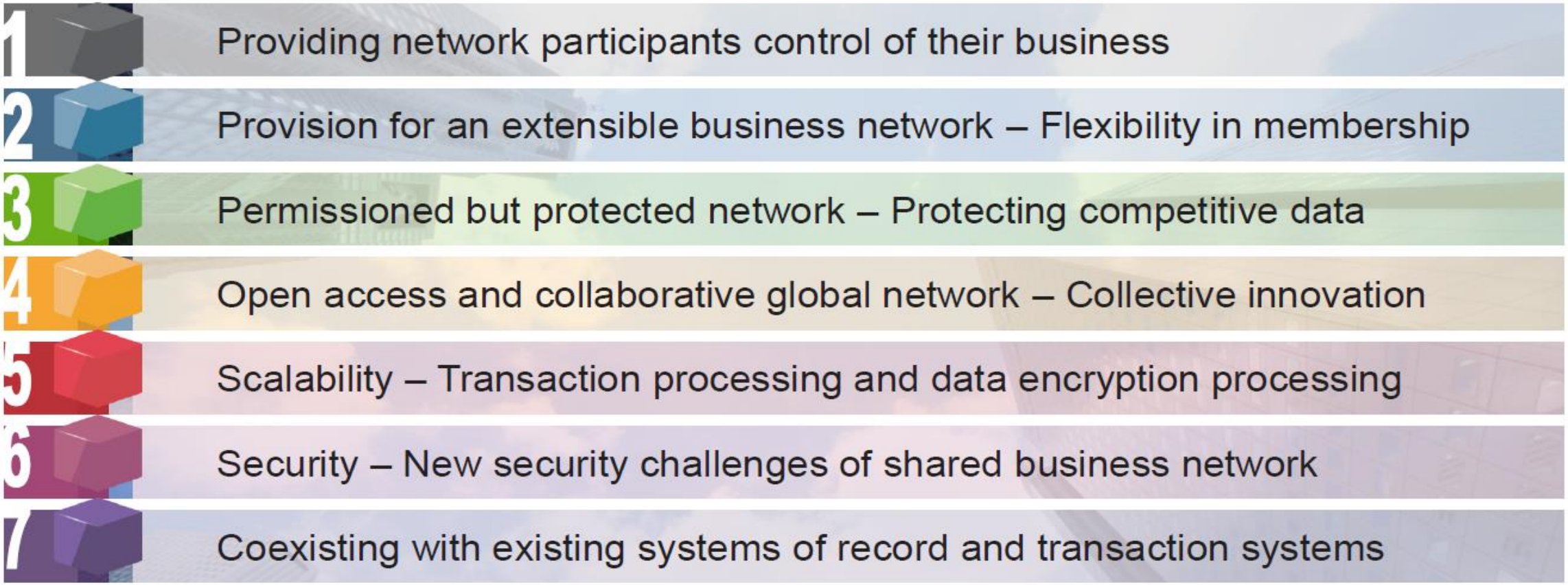
Hyperledger Fabric

- Components: Consensus, membership services.
- Leverages container technology to host smart-contract called chain-code which holds the application logic.

Hyperledger Fabric Functionality

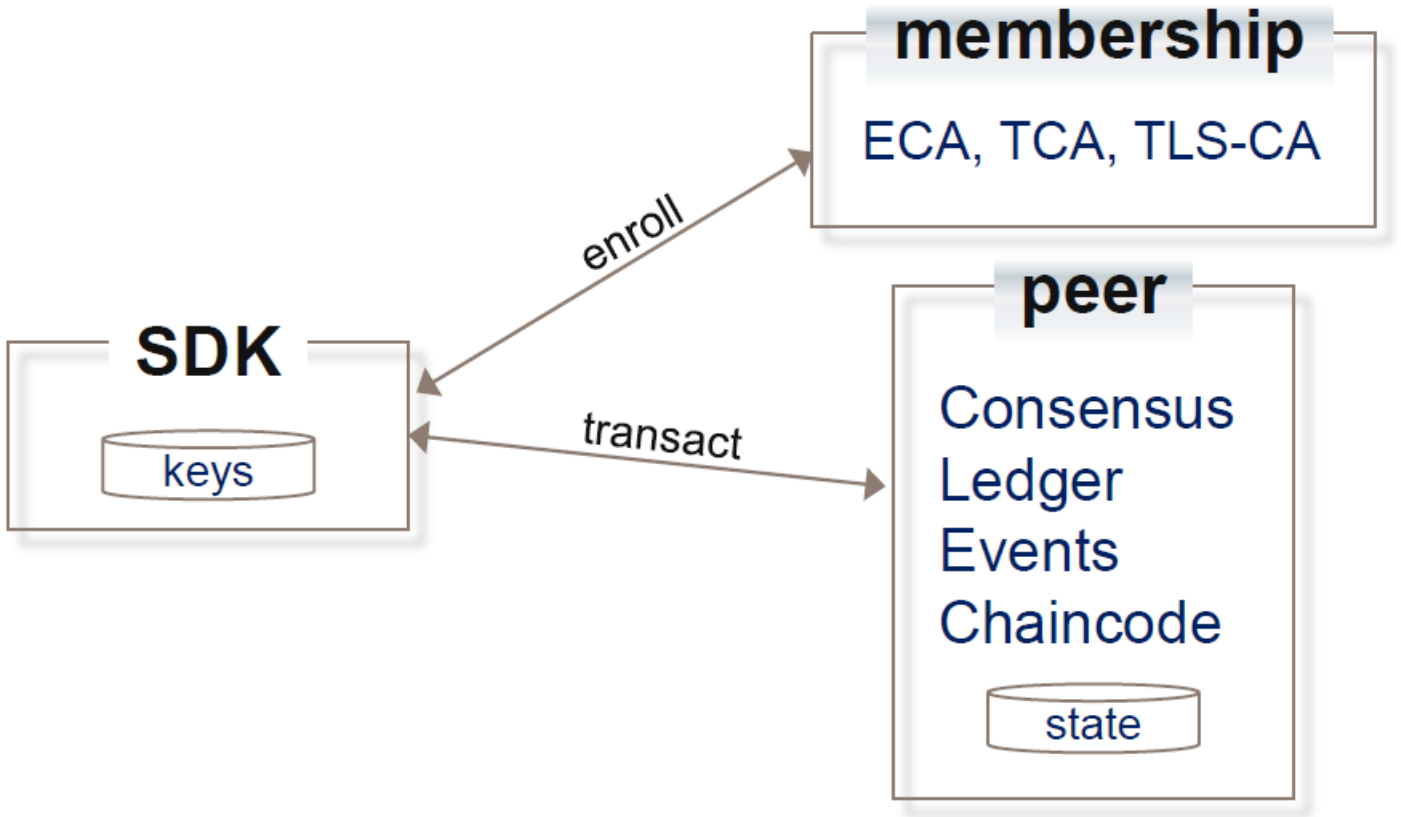
- Identity Management
- Privacy and confidentiality
- Efficient processing
- Chain-code functionality
- Modular Design

Seven design principles of sustainable Blockchain business networks



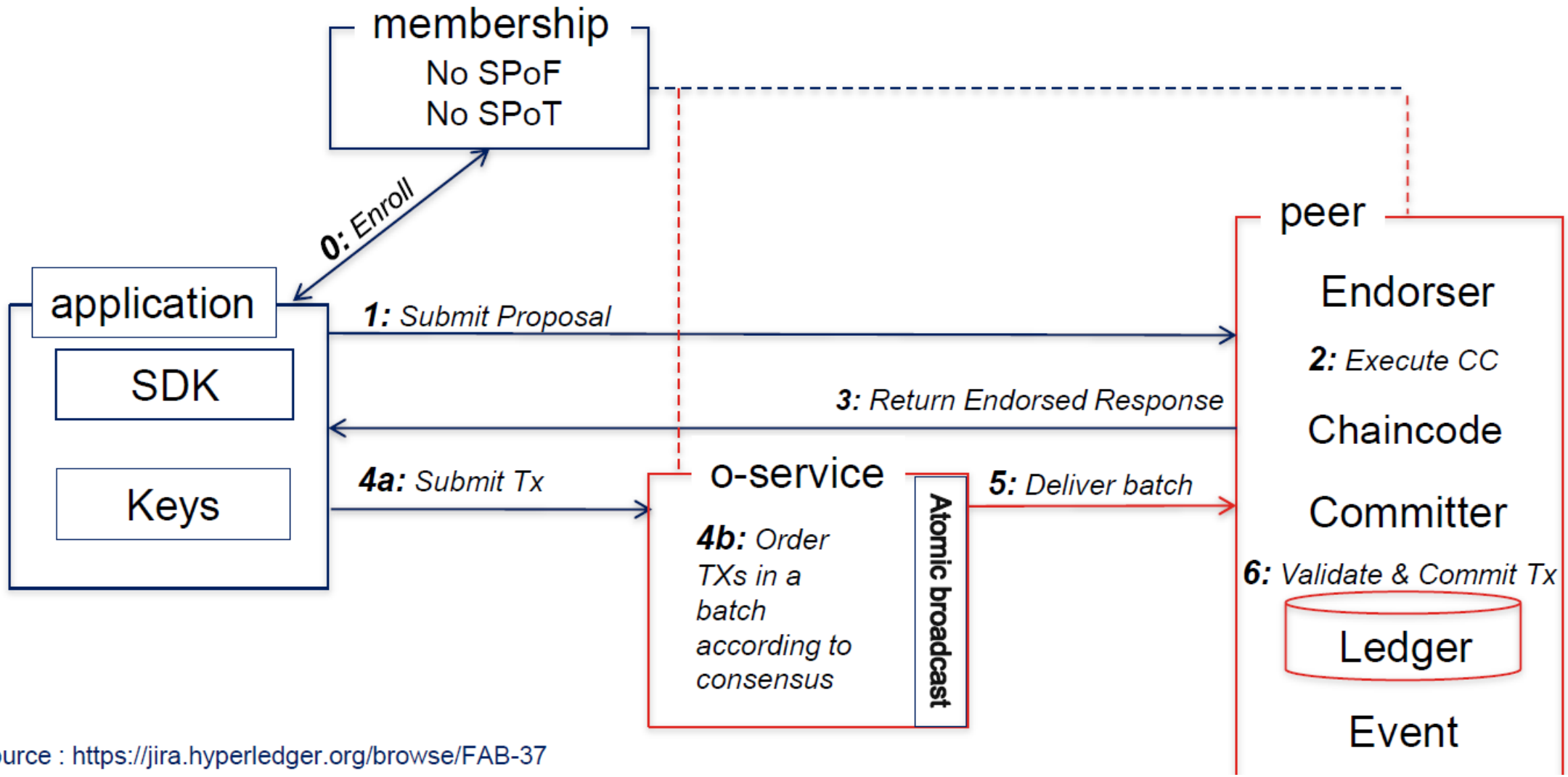
Slides Courtesy: IBM Corporation

Architecture of Hyperledger Fabric v0.6



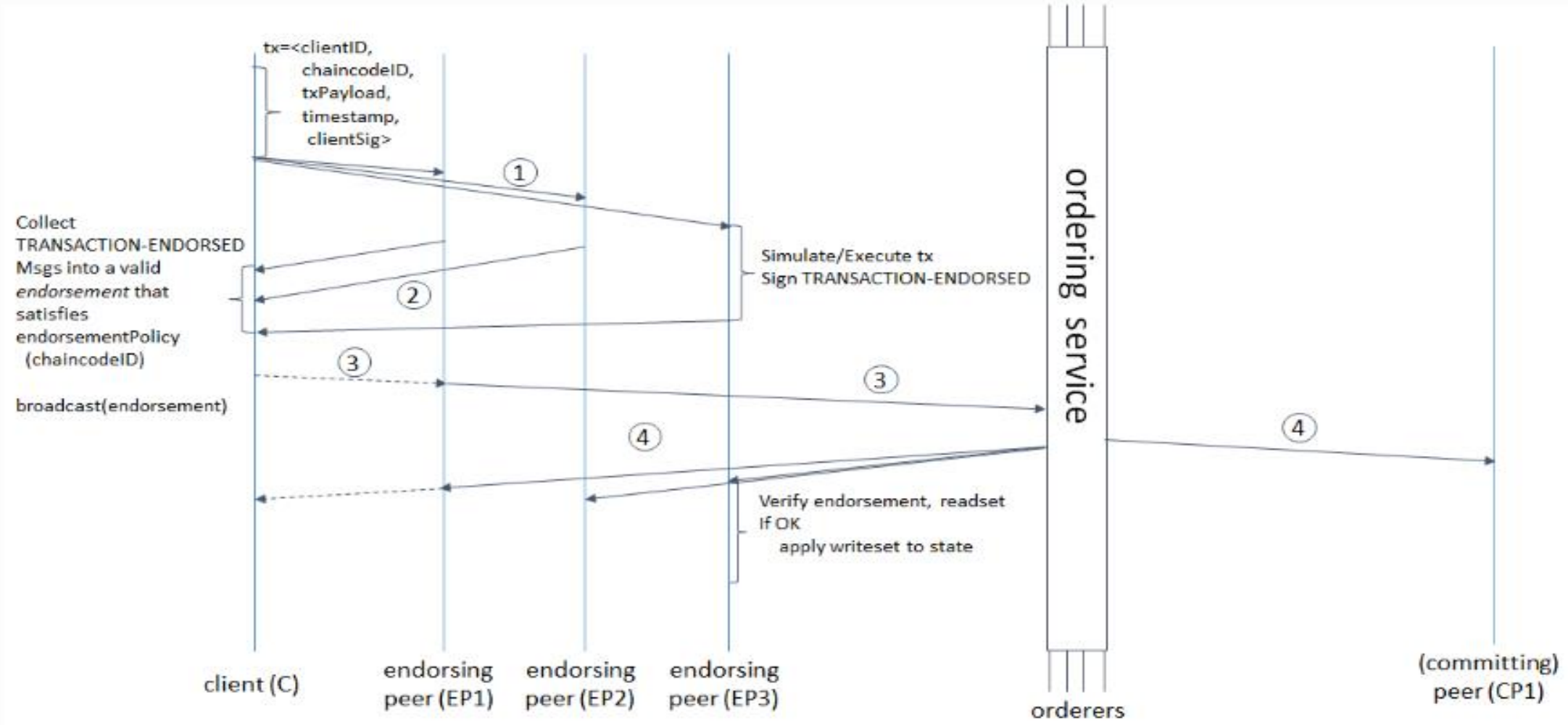
Slides Courtesy: IBM Corporation

Architecture of Hyperledger Fabric v1



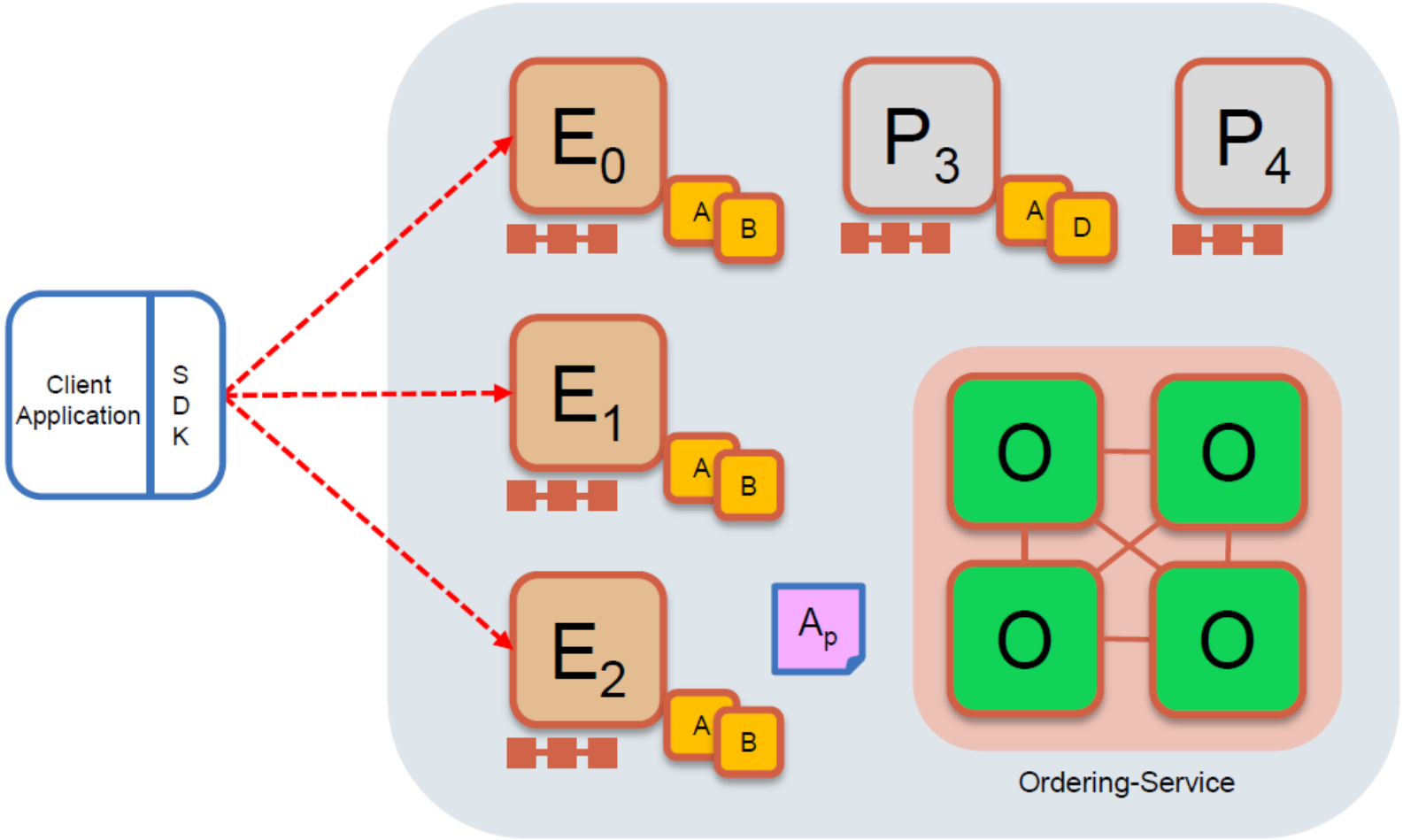
Source : https://jira.hyperledger.org/browse/FAB-37

Transaction Flow



Source: Hyperledger Fabric Website

Sample transaction: Step 1/7 – Propose transaction

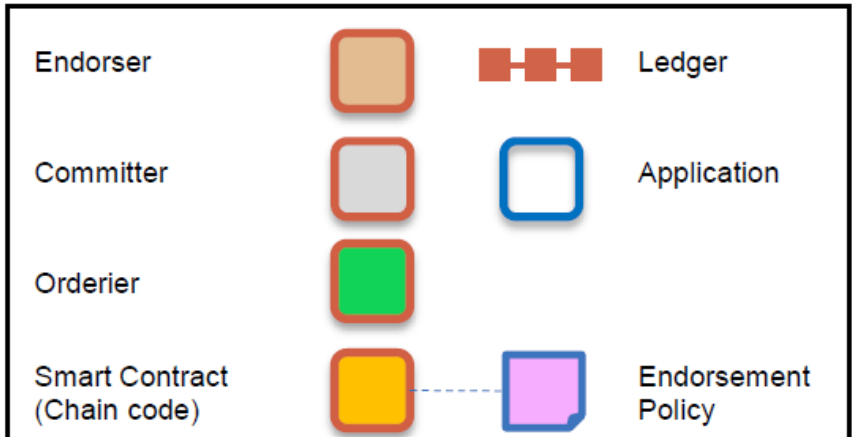


Application proposes transaction

Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

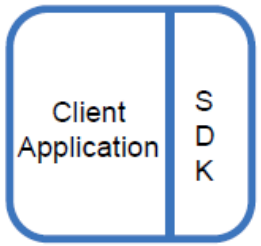
Client application submits a transaction proposal for **chaincode A.** It must target the required peers {E_0, E_1, E_2}

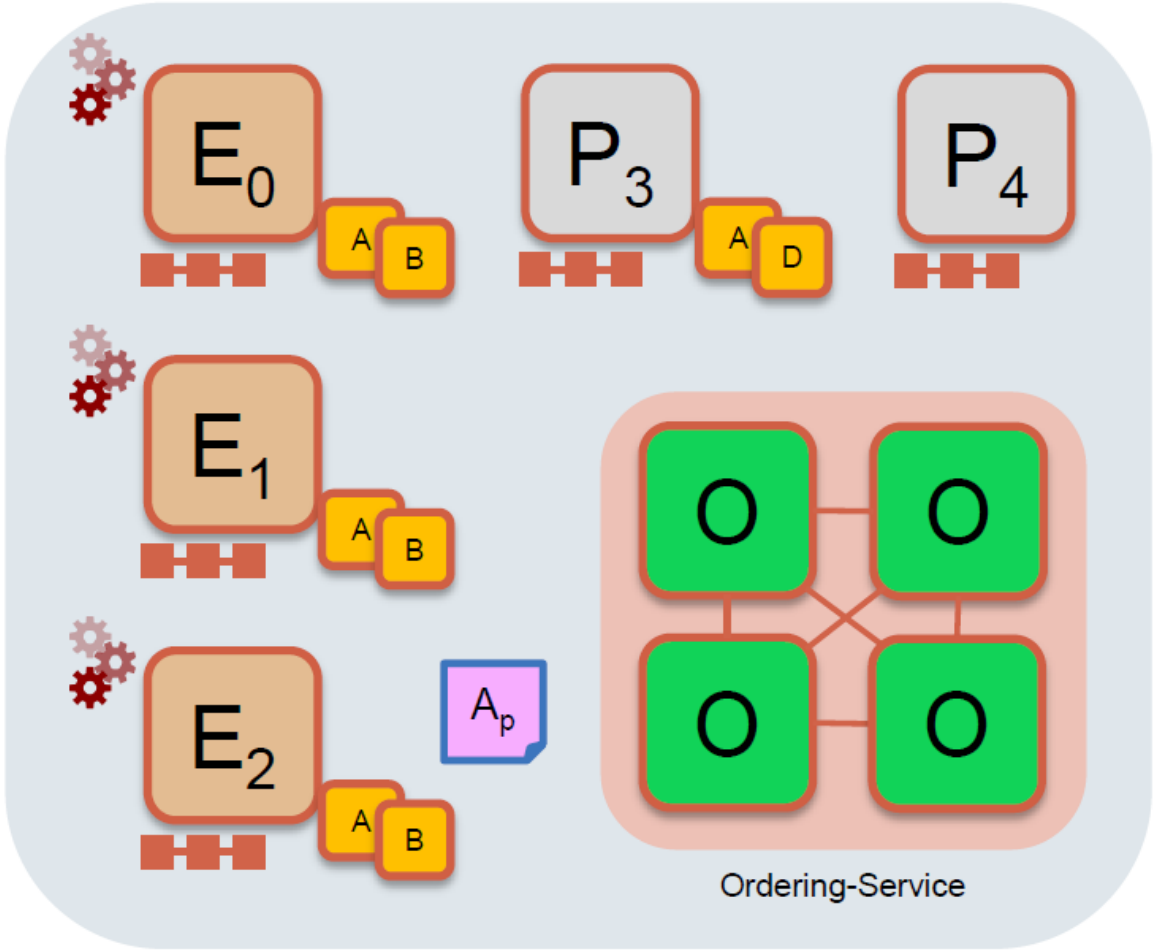
Key:



Slides Courtesy: IBM Corporation

Sample transaction: Step 2/7 – Execute proposal



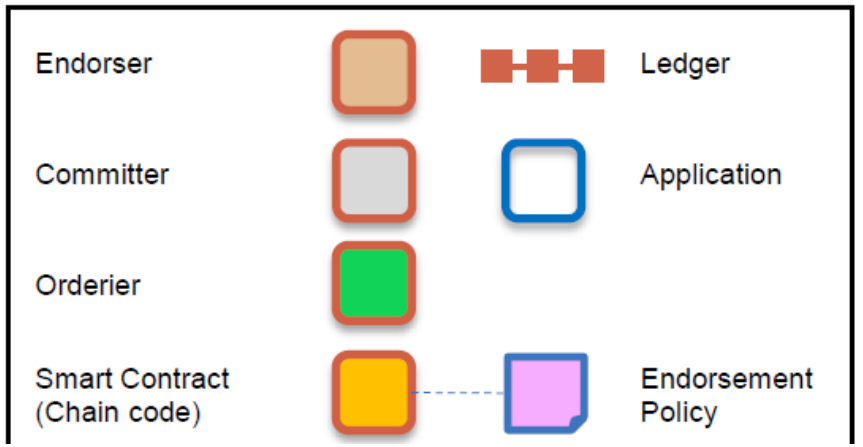


Endorsers Execute Proposals

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

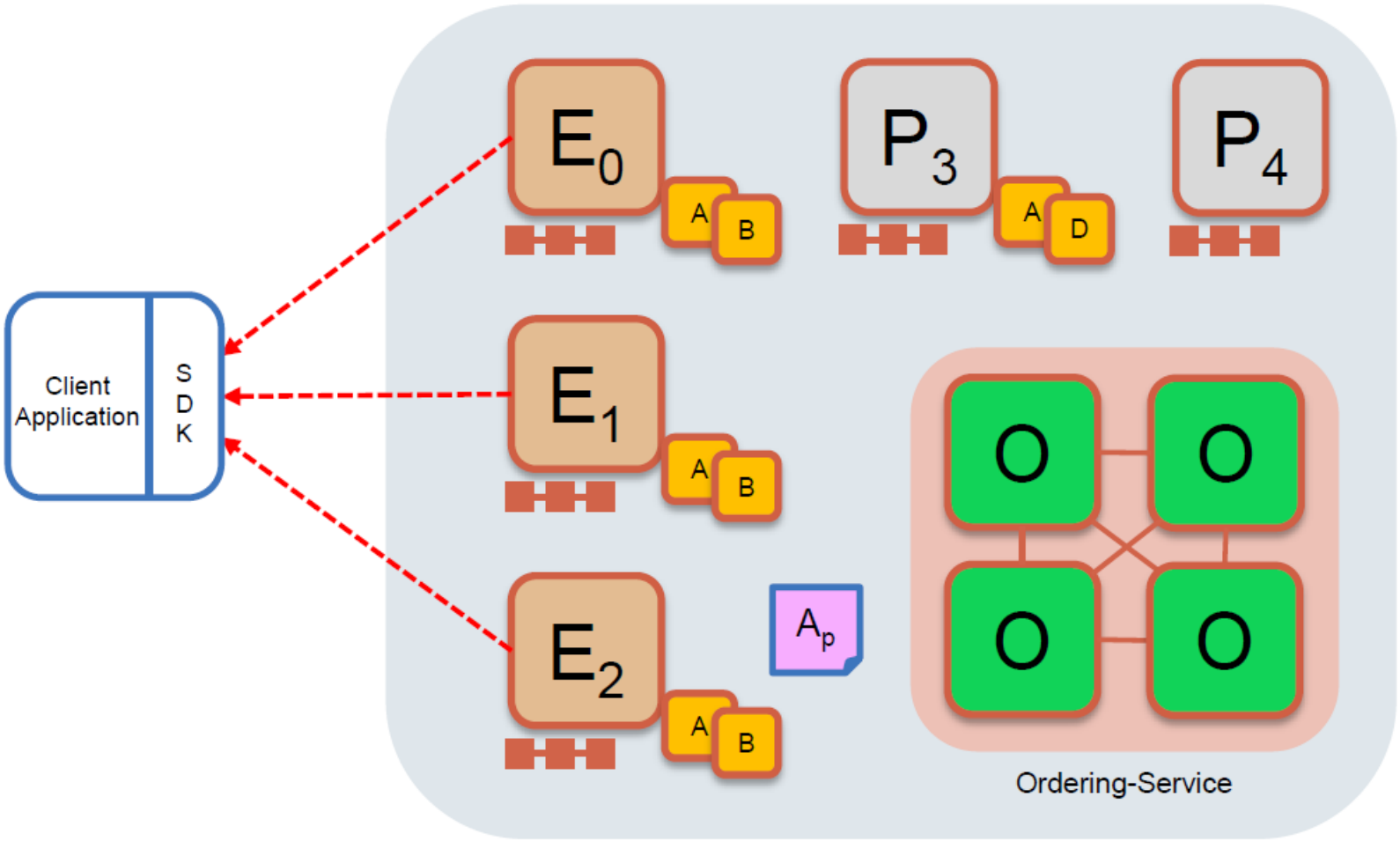
Each execution will capture the set of **R**ead and **W**ritten data, called **RW sets,** which will now flow in the fabric.

Key:



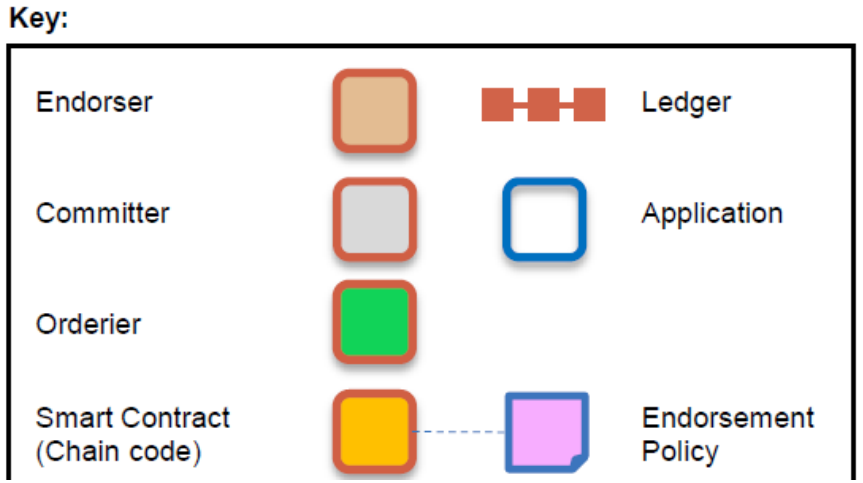
Slides Courtesy: IBM Corporation

Sample transaction: Step 3/7 – Proposal Response



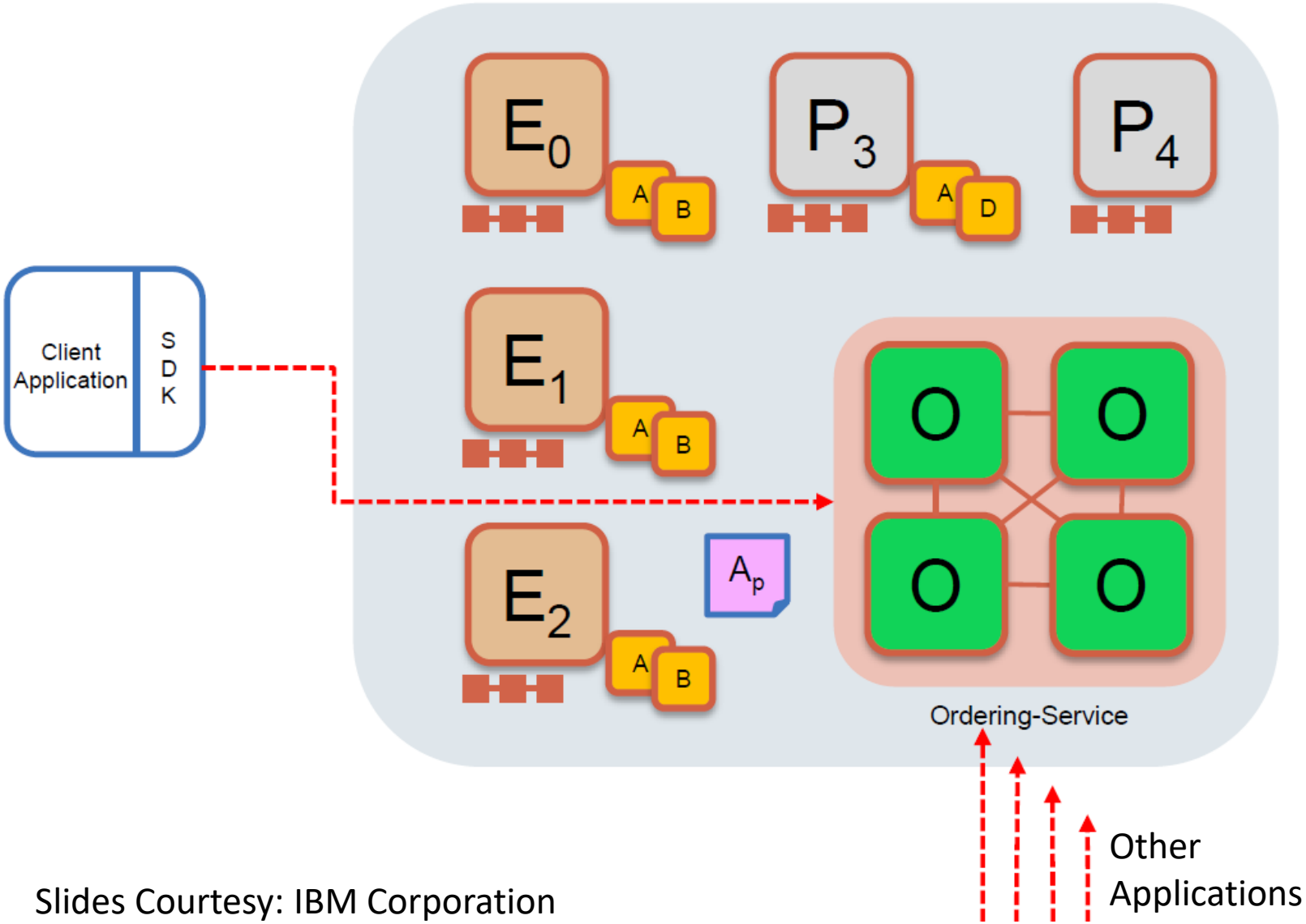
Application receives responses

The RW sets are signed by each endorser and returned to the application



Slides Courtesy: IBM Corporation

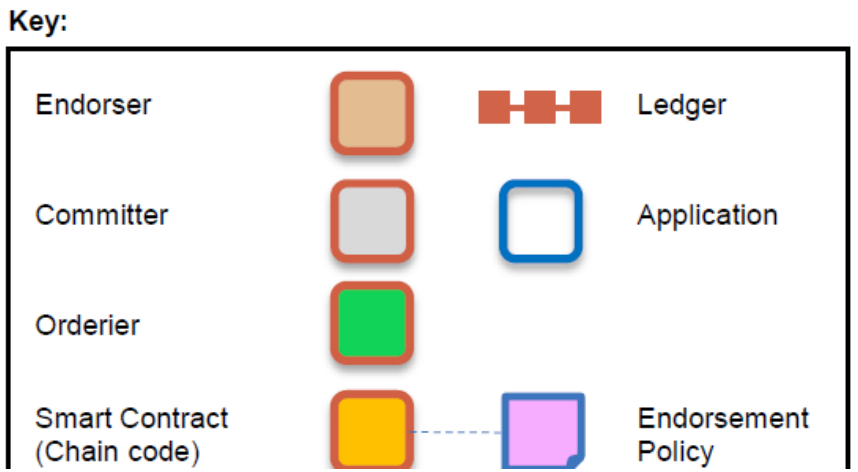
Sample transaction: Step 4/7 – Order Transaction



Application submits responses for ordering

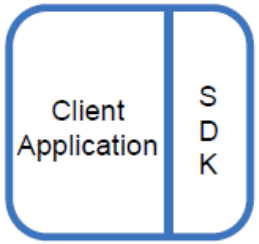
Application submits responses as a **transaction** to be ordered.

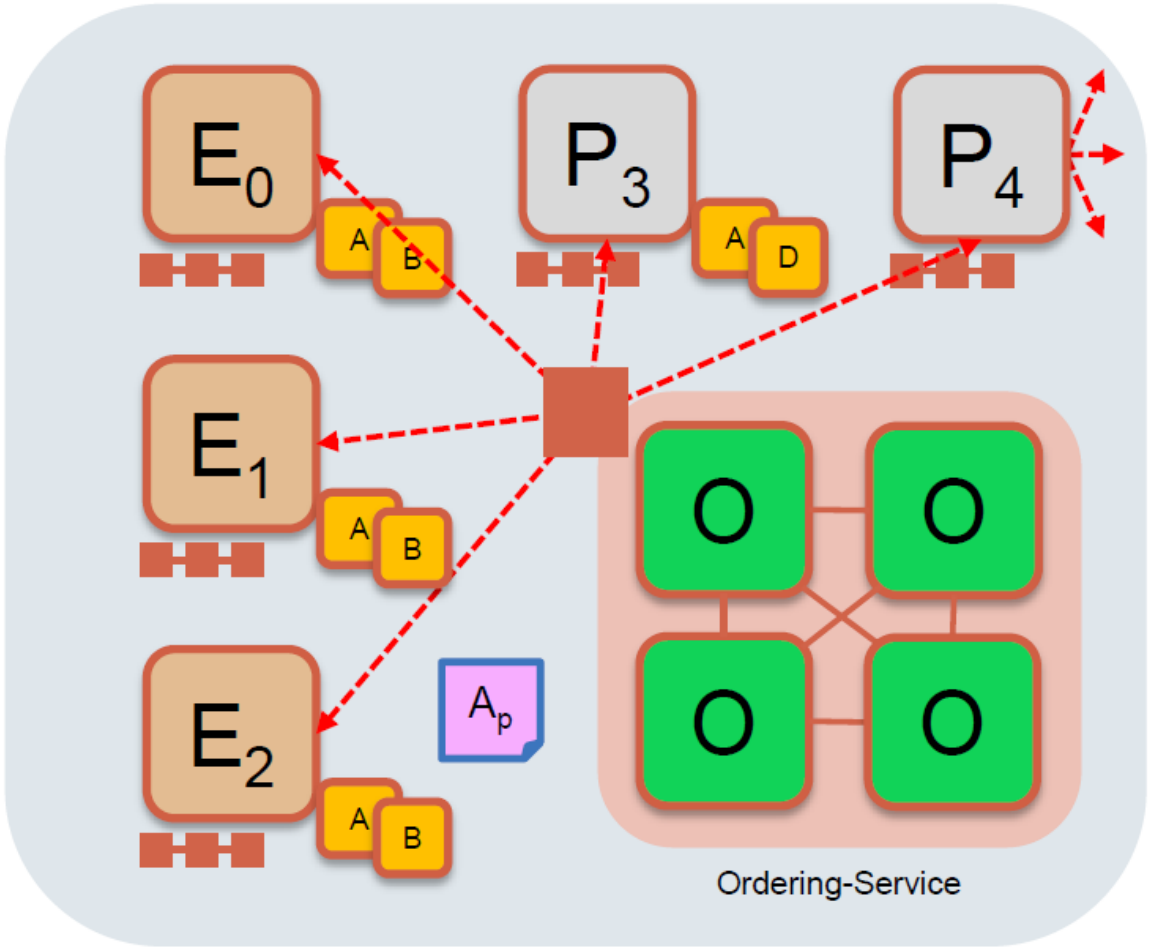
Ordering happens across the fabric in parallel with transactions submitted by other applications



Slides Courtesy: IBM Corporation

Sample transaction: Step 5/7 – Deliver Transaction





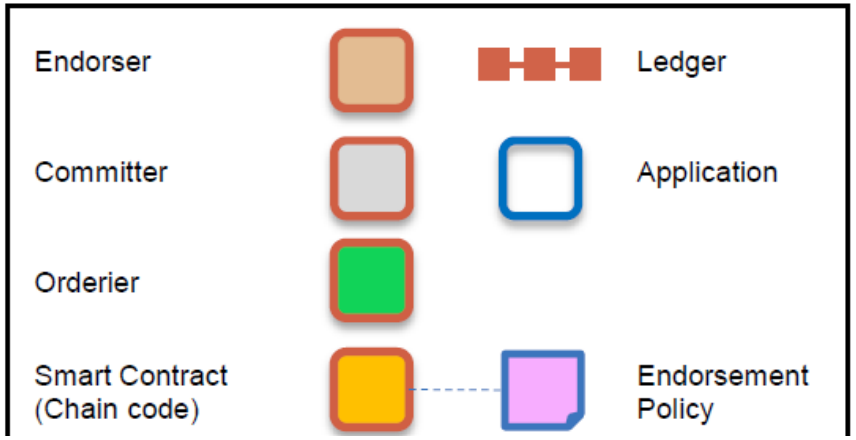
Orderer delivers to all committing peers

Ordering service collects transactions into blocks for distribution to committing peers. Peers can deliver to other peers using gossip (not shown)

Different ordering algorithms available:

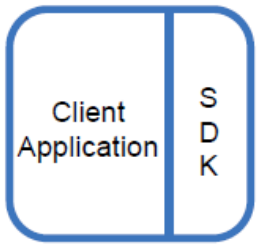
- SOLO (single node, development)
- Kafka (blocks map to topics)
- SBFT (tolerates faulty peers, future)

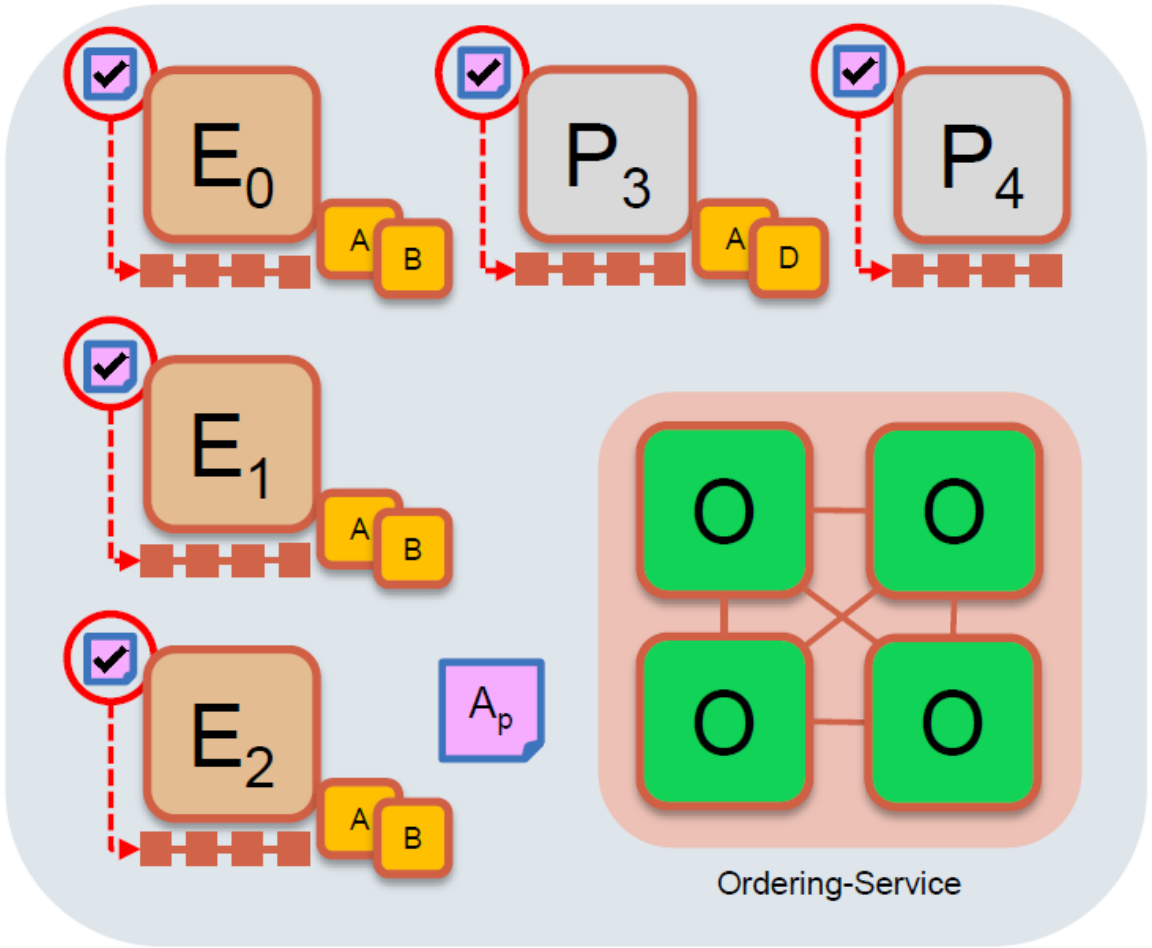
Key:



Slides Courtesy: IBM Corporation

Sample transaction: Step 6/7 – Validate Transaction



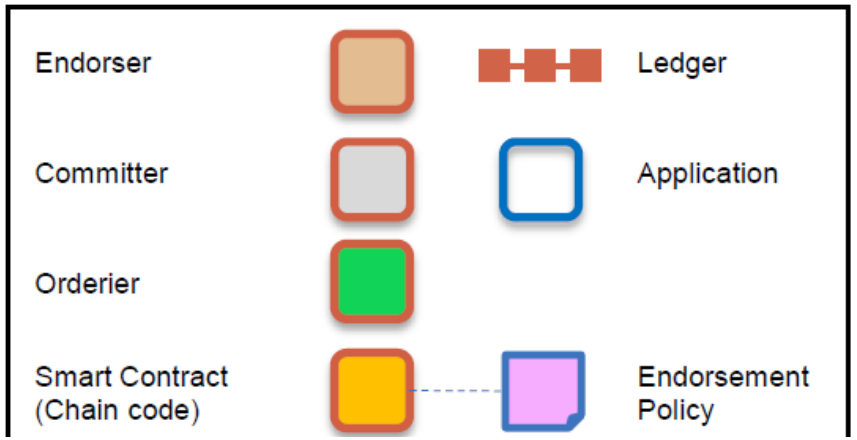


Committing peers validate transactions

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

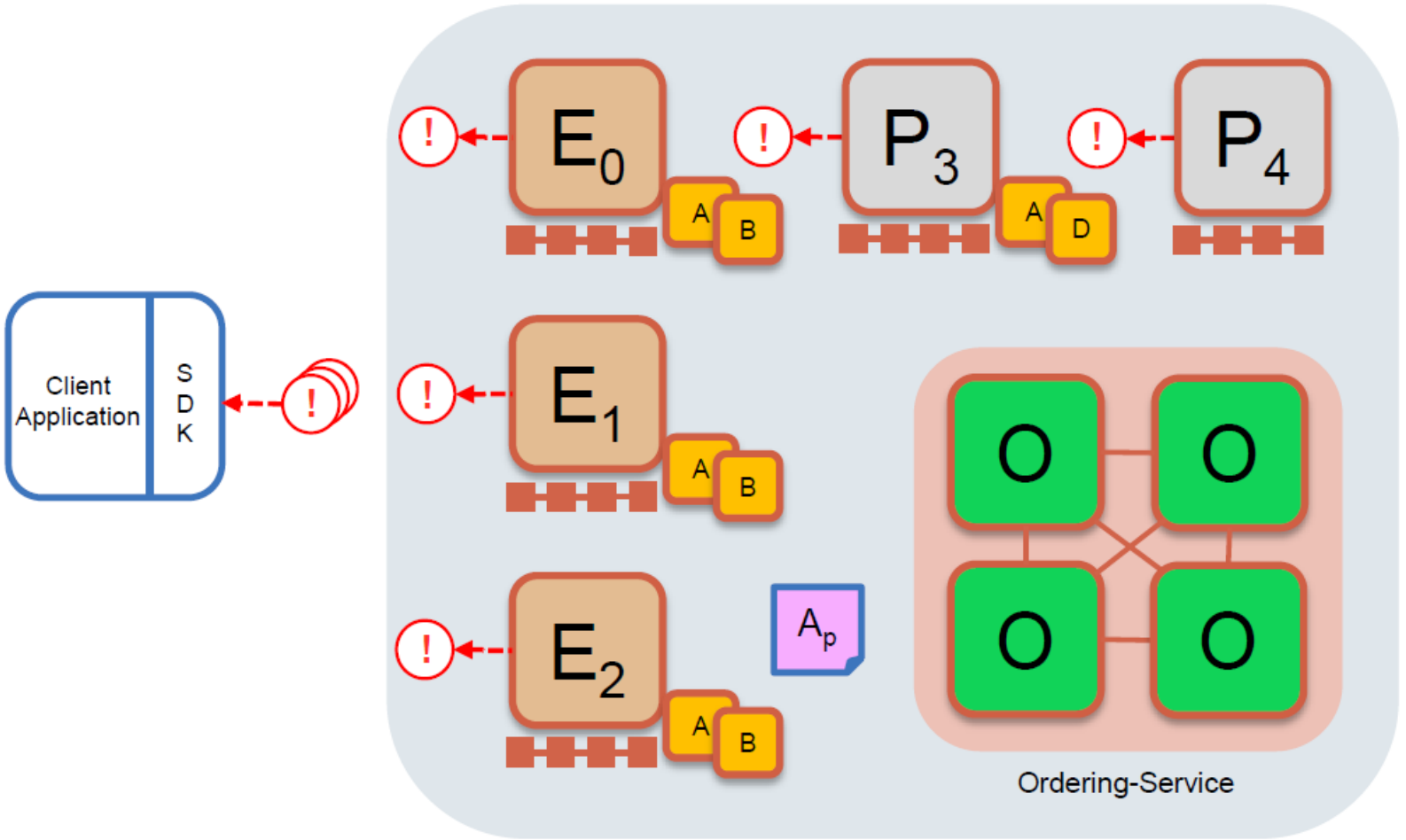
Transactions are written to the ledger and update caching DBs with validated transactions

Key:



Slides Courtesy: IBM Corporation

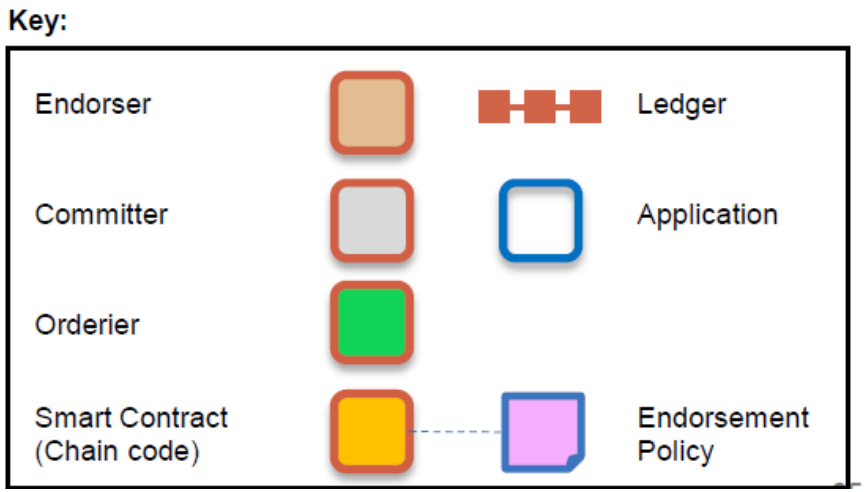
Sample transaction: Step 7/7 – Notify Transaction



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



Slides Courtesy: IBM Corporation

Disadvantages of Using Blockchain

- High energy consumption on the public blockchain mining
- Several attacks including centralization of hashing power problem
- Repetition of the same data on all the nodes
- Speculative market
- Network speed/cost
- Size of the block
- Hard and soft fork
- Immutable smart contract

Source: https://medium.com/nudjed/blockchain-advantage-and-disadvantages-e76dfde3bbc0

Advantages of Using Blockchain

- No need to trust on the centralized authority!
- Elimination of intermediaries
- Real time settlements
- Reduced operational cost
- High level of transparency
- Distributed
- Decentralized
- Immutable data

References

1. Ethereum Platform: https://www.ethereum.org/
2. Smart-Contract Tutorial: https://solidity.readthedocs.io/en/v0.4.24/
3. How to create private Ethereum Blockchain: https://medium.com/mercuryprotocol/how-to-create-your-own-private-ethereum-blockchain-dad6af82fc9f
4. Bitcoin Whitepaper: https://bitcoin.org/bitcoin.pdf
5. Blockchain Blog: https://blockgeeks.com

References on Healthcare Security

1. https://www.cisecurity.org/cyber-attacks-in-the-healthcare-sector/
2. http://www.healthcareitnews.com/slideshow/biggest-healthcare-breaches-2017-so-far?page=1
3. http://theconversation.com/why-has-healthcare-become-such-a-target-for-cyber-attackers-80656
4. http://fortune.com/2017/05/15/ransomware-attack-healthcare/
5. https://nakedsecurity.sophos.com/2016/04/26/why-cybercriminals-attack-healthcare-more-than-any-other-industry/
6. https://www-03.ibm.com/press/us/en/pressrelease/51394.wss
7. MedRec: https://viral.pubpub.org/pub/medrec
8. G. Zyskind et al.,“Decentralizing Privacy: Using Blockchain to protect protect personal data”

Questions?

Thank You