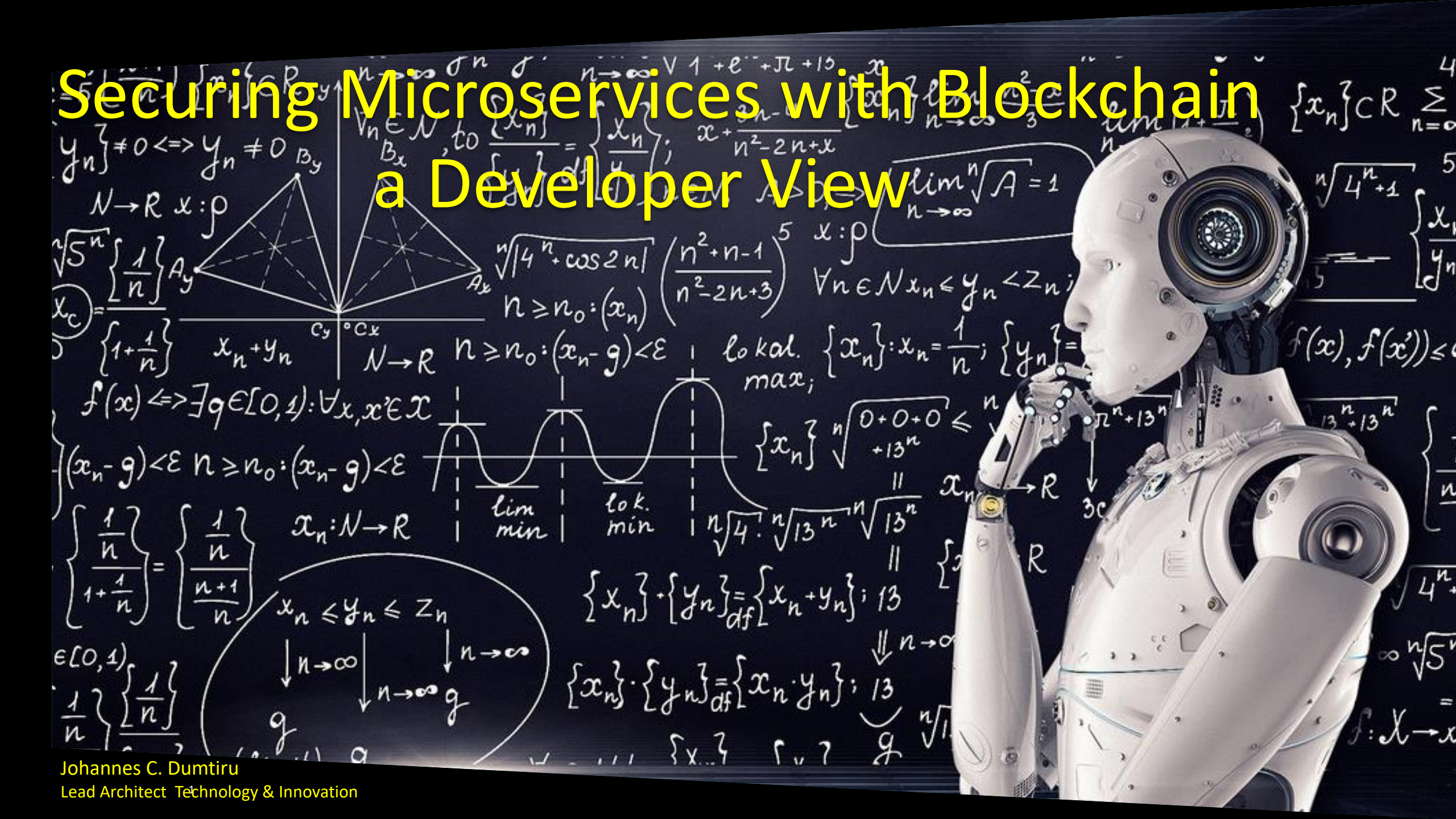
Securing Microservices with Blockchain
a Developer View



Johannes C. Dumtiru
Lead Architect Technology & Innovation

Agenda

Motivation/ Context

Future of Distributed Applications - Innovation

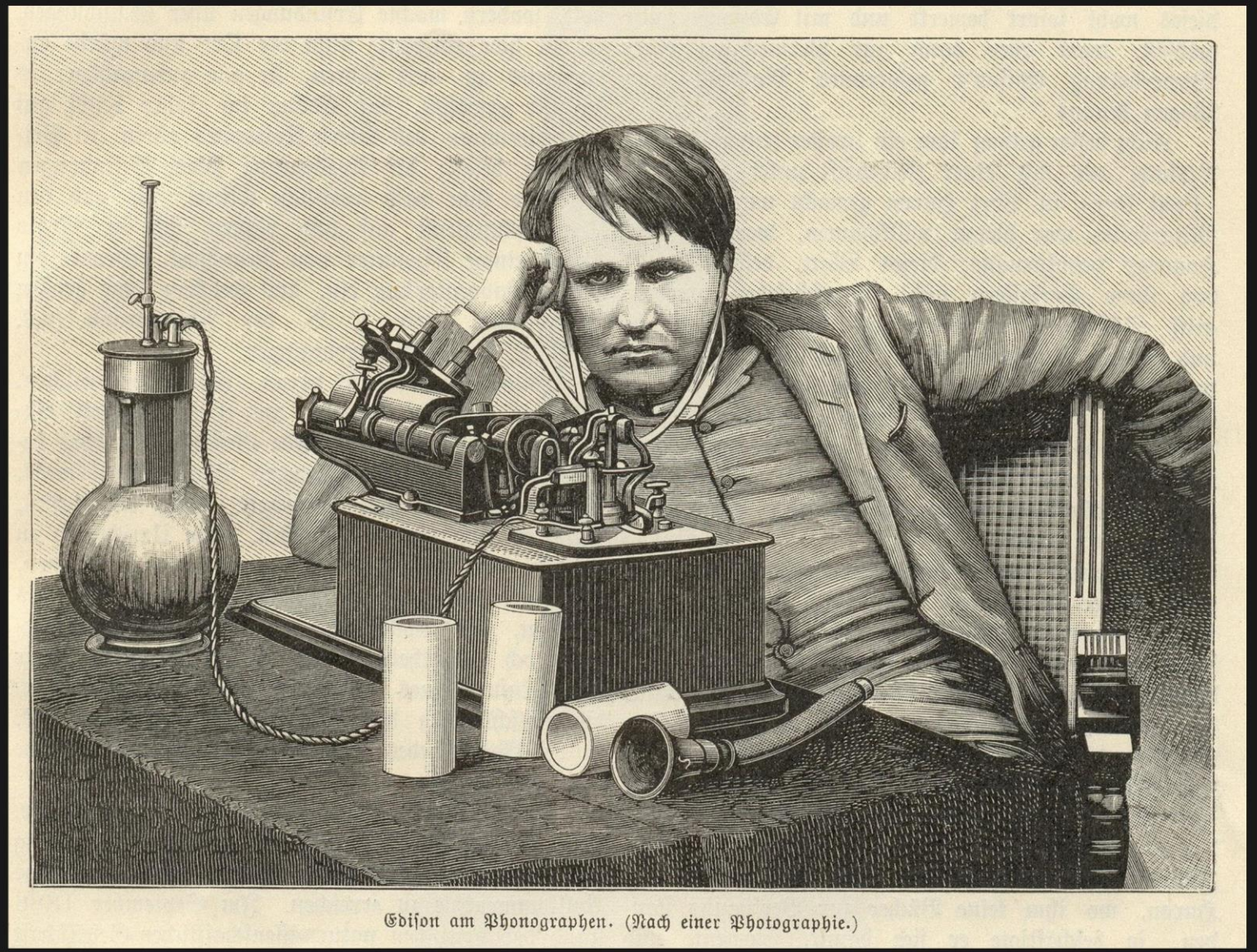
Blockchain - Microservice analogy basics

- Architecture Patterns
- Implementation samples

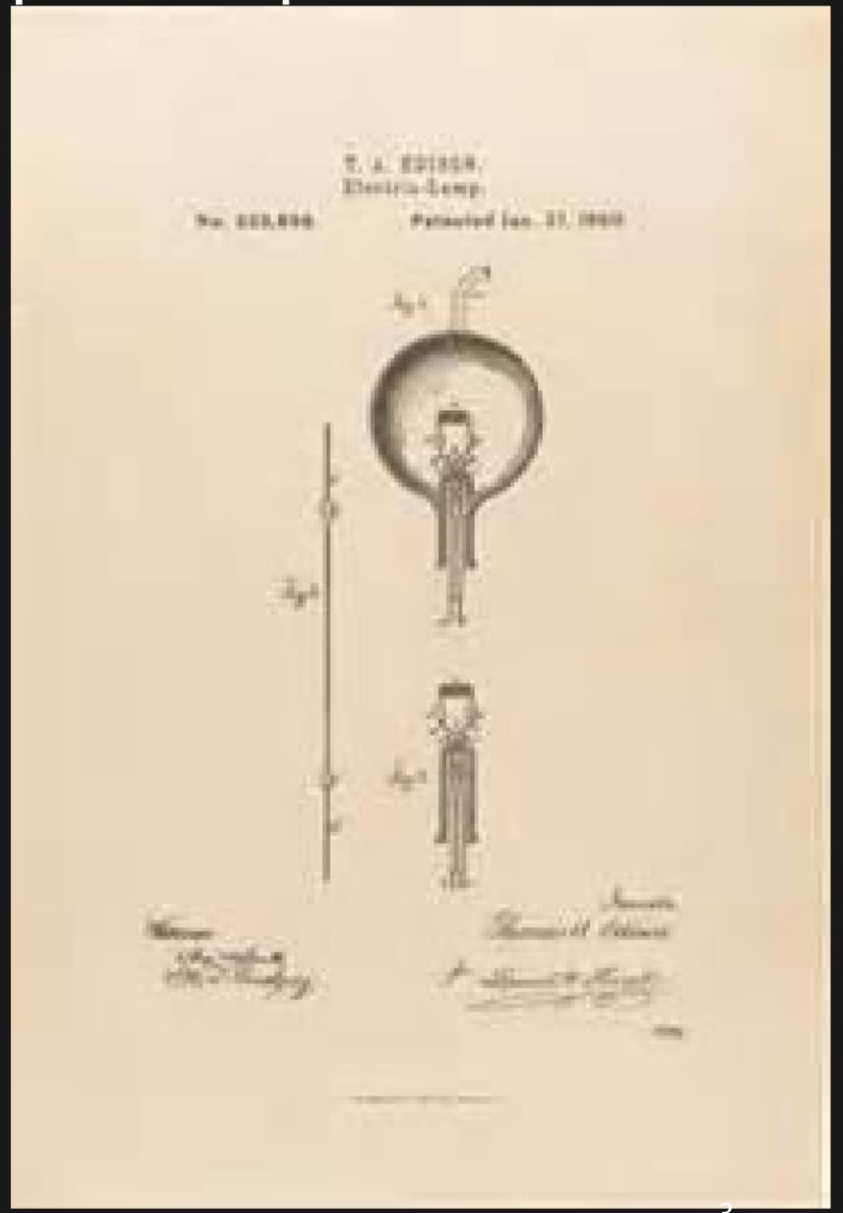
Blockchain UsesCases

- Blockhain as Game Changer in Industry
- How to get started

Edison started as a telegraph operator



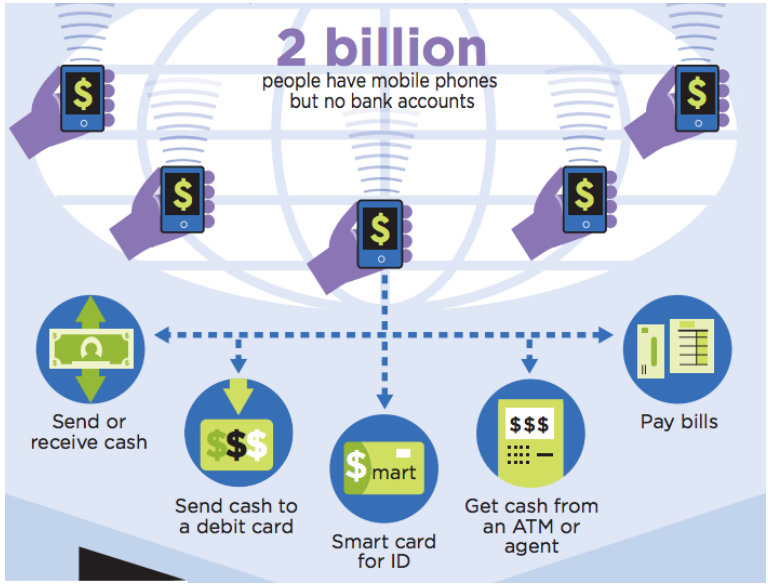
Edison am Phonographen. (Nach einer Photographie.)

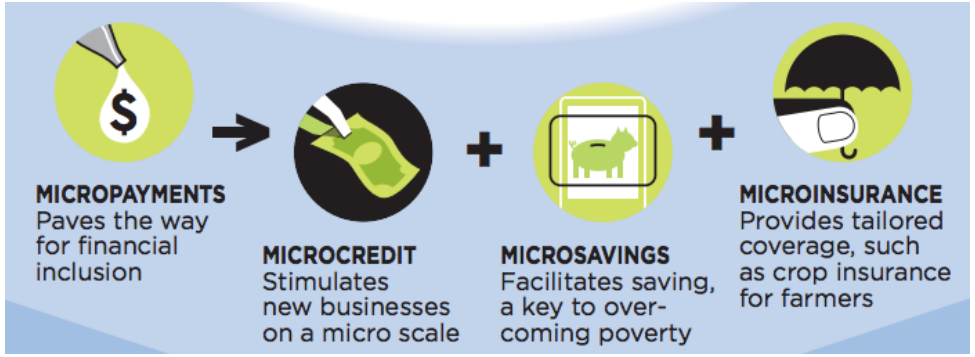


It's about Changing the way of thinking about Data

Changing R&D

Changing finance





Changing ERP & CRM

Real World Problem: Counterfeit Pharmaceuticals

10%-30% of drugs sold in developing countries are **counterfeits**.

80% of the counterfeit drugs consumed in the US come from overseas.

Prescription drugs market is estimated at **$900 billion worldwide**.

The global market for counterfeit drugs is **$200 billion** dollars.

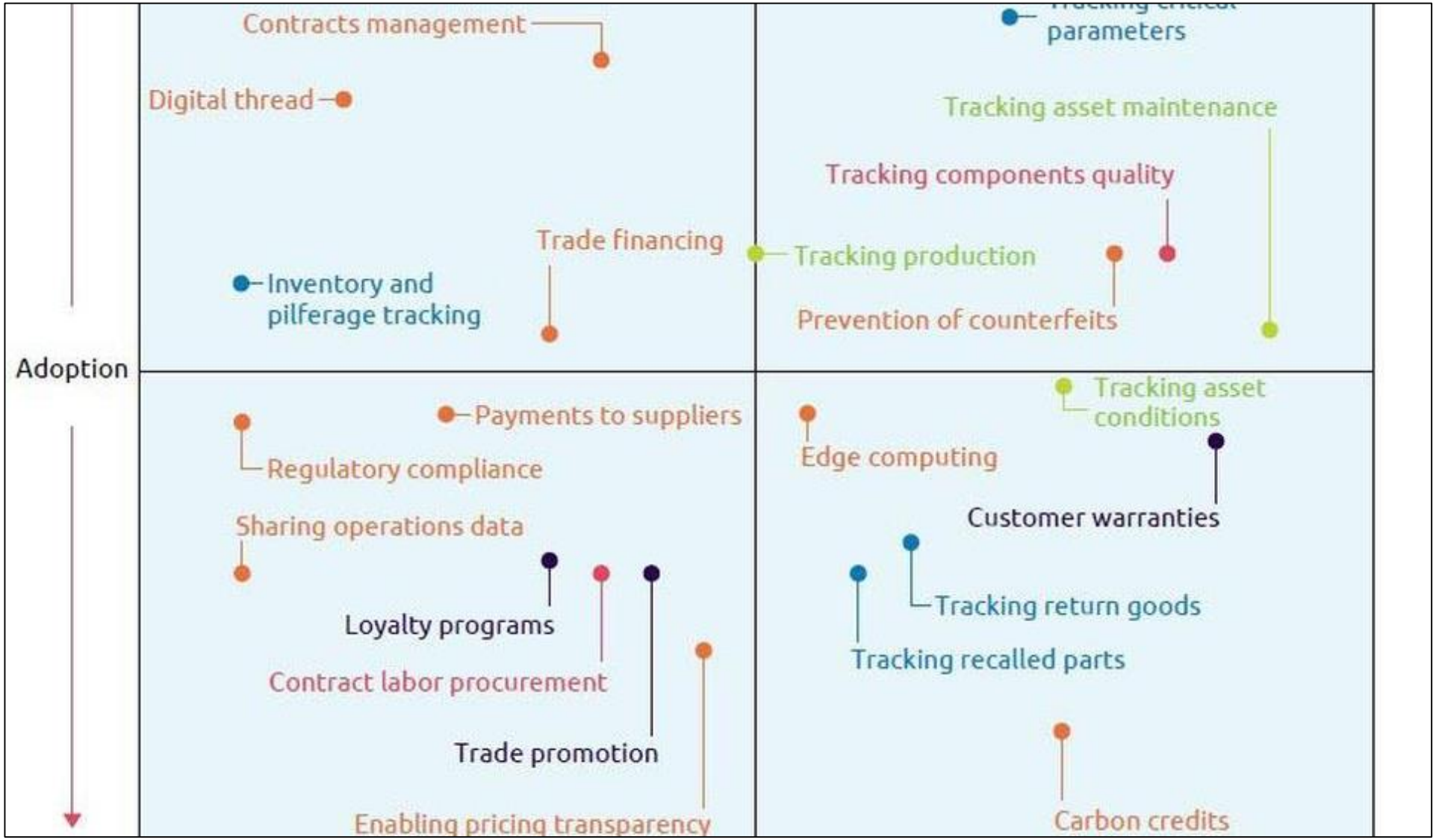
The global market for counterfeit products is **$460 billion** dollars.

More than **30%** of counterfeit drugs **do not contain** any active ingredients.

Changing Needs

http://www.bbc.com/future/story/20160504-we-looked-inside-a-secret--bitcoin-mine

Adaption



Gartner

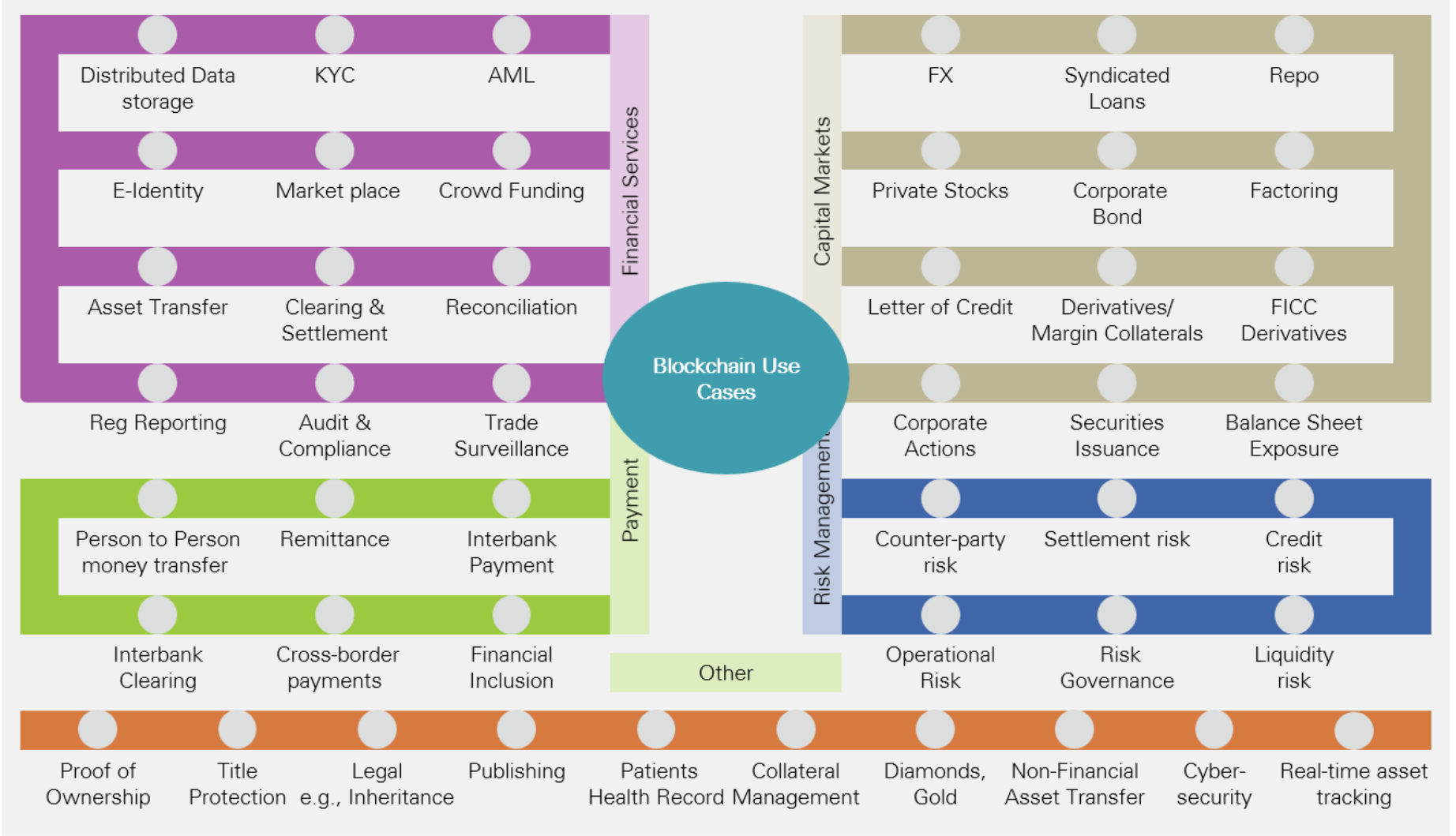
Disruptive to all industries…

- Blockchain can flatten ecosystems and supply chains, removing middlemen processes (companies or divisions)
- Peer-to-Peer value exchange, dramatically reduce settlement time
- Fundamentally changes Audit at the DNA level, regulator output a byproduct
- Introduces new distributed or shared economics – a collaborative economy
  - Shared incentives, costs, risk
- Could revolutionize identity…

Fear
Derive revenue by brokering or settling transactions between counterparties – Uber, Paypal, Swift, DTCC, Correspondent Banking (x-boarder payments)…
Derive revenue from existing friction in supply chains
Arbitrage on opacity of markets

Greed
Cut out middleman brokering trust to counterparties
Consume cross industry workloads – improving service and margin – manufacture finance on steroids, hospital network insurance policy

Changing transactions UseCases



Report: Blockchain Could Disrupt Capital Markets Within Decade

Yessi Bello Perez (@yessi_kbello) | Published on November 10, 2015 at 21:39 GMT

NEWS

From hype to understanding of essence

Its all about trust…

between ourselves or “us”…

to transform how we exchange value
following common rules with each other …

establishing a shared truth based on
cryptographic proofs…

to revolutionize the execution of
contractual agreements between us…

forever.

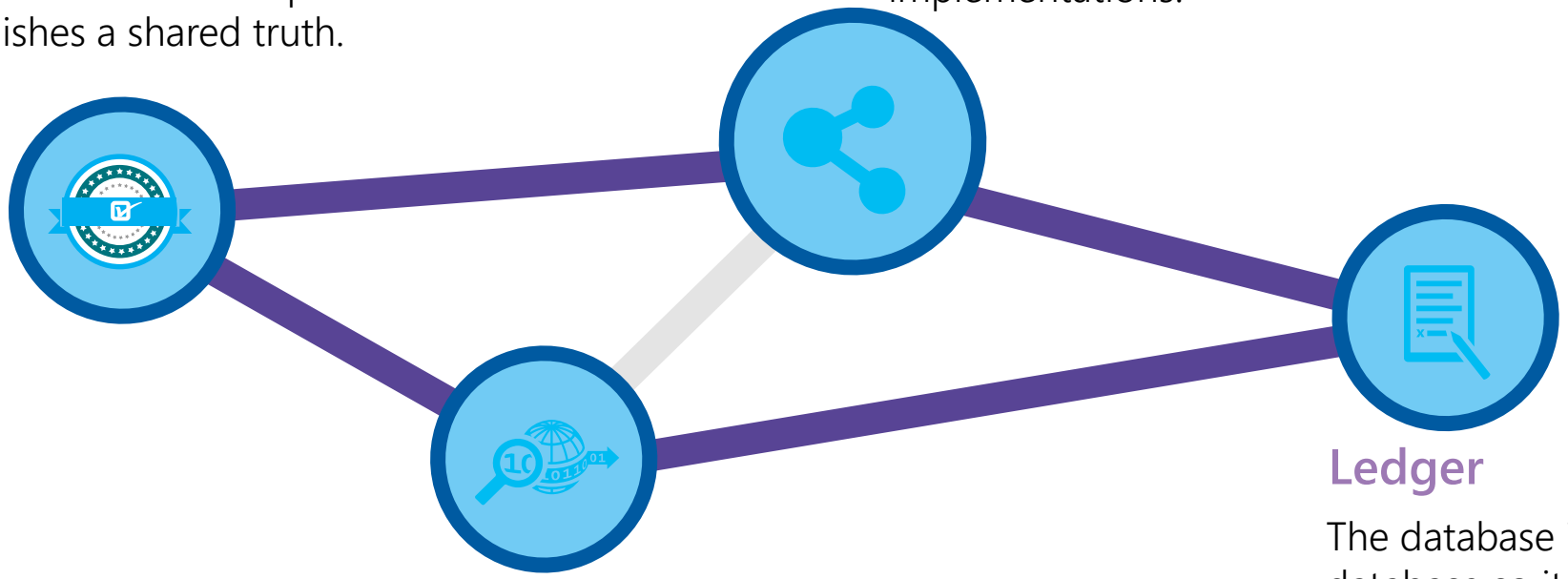
Blockchain or Distributed Ledger | 4 Pillars

Cryptographically Authentic

Uses tried and true public/ private signature technology. Blockchain applies this technology to create transactions that are impervious to fraud and establishes a shared truth.

Shared

Blockchain's value is directly linked to the number of organizations or companies that participate in them. There is huge value for even the fiercest of competitors to participate with each other in these shared database implementations.



Distributed

There are many replicas of the Blockchain database. In fact, the more replicas there are, the more authentic it becomes.

Ledger

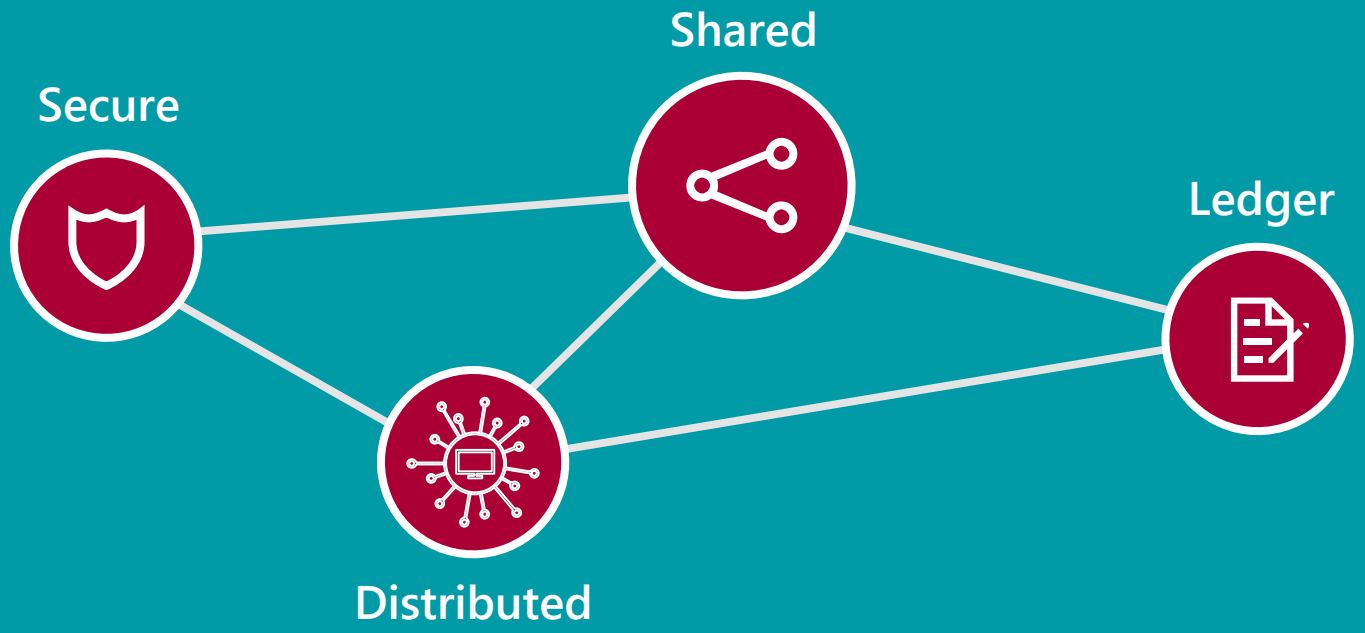
The database is a read/write-once database so it is an immutable record of every transaction that occurs.

Blockchains

Cryptocurrency is built on *blockchain*

Blockchains are also called Distributed Ledger Technology (DLT)

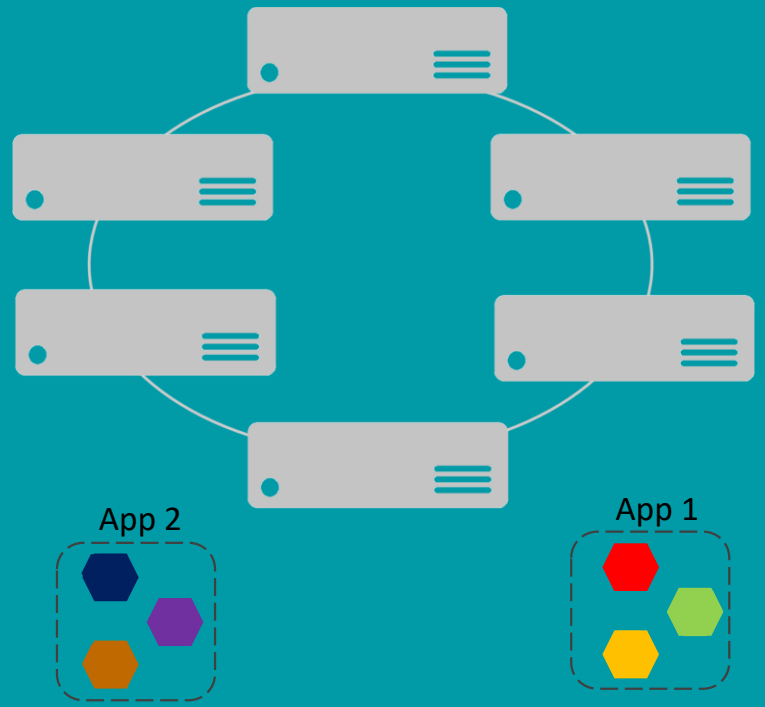
Blockchains gets their properties from cryptography



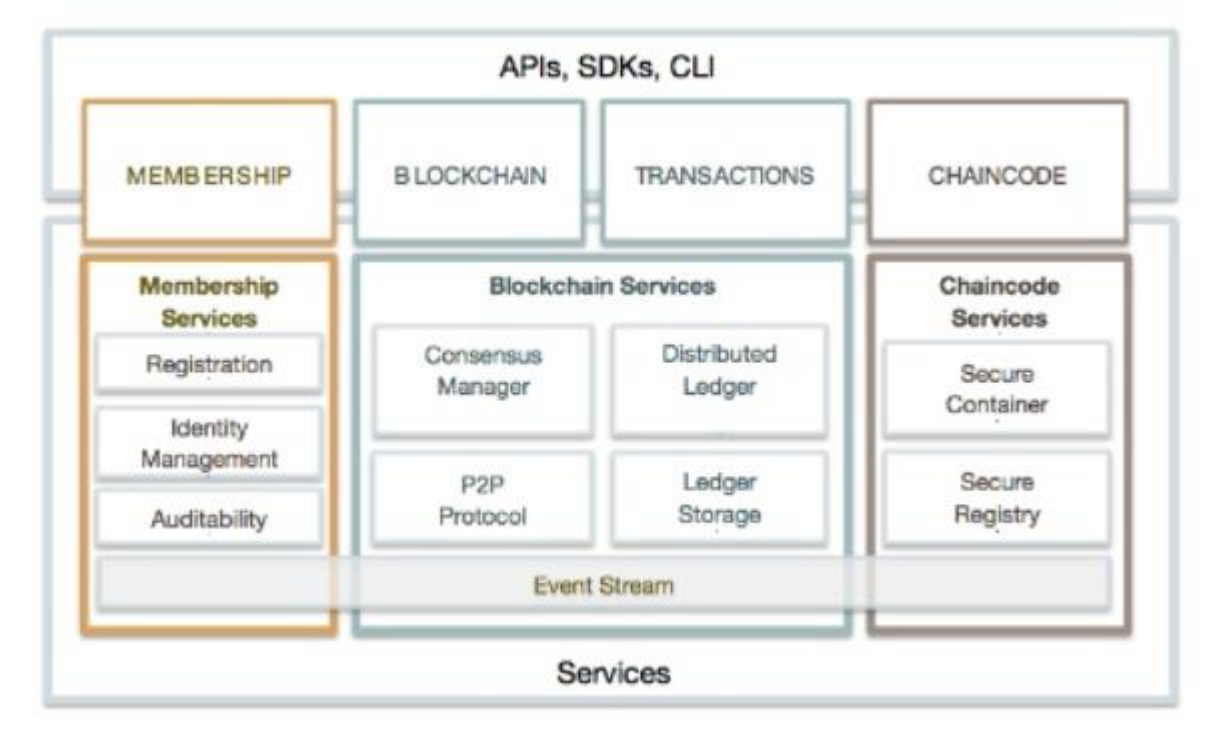
Microservices

application approach

- A microservice application separates functionality into distributed smaller services.
- Scales out by deploying each service independently creating instances of these services across servers/VMs/containers

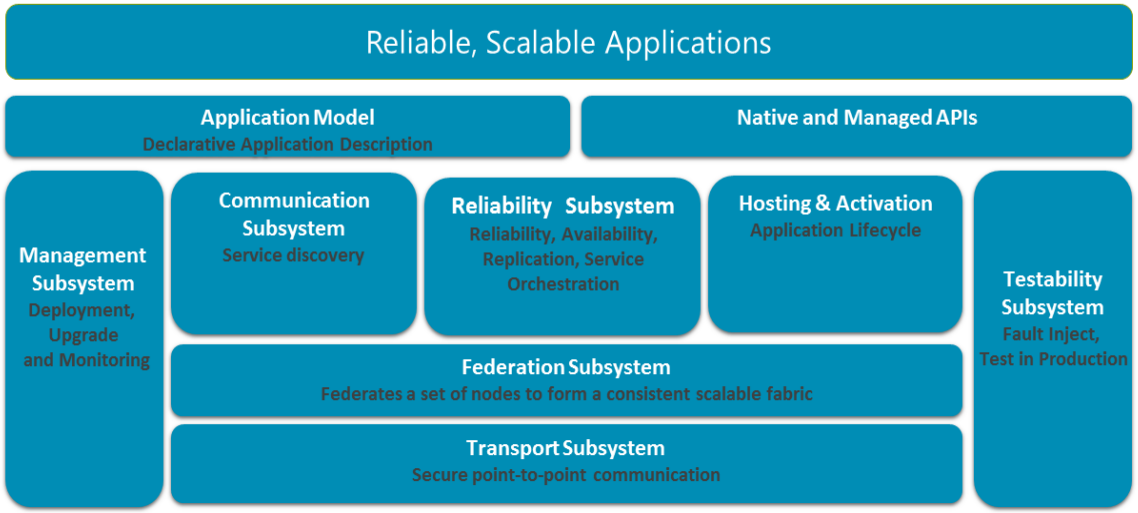


An Architectural View



Hyperledger fabric

Service Fabric



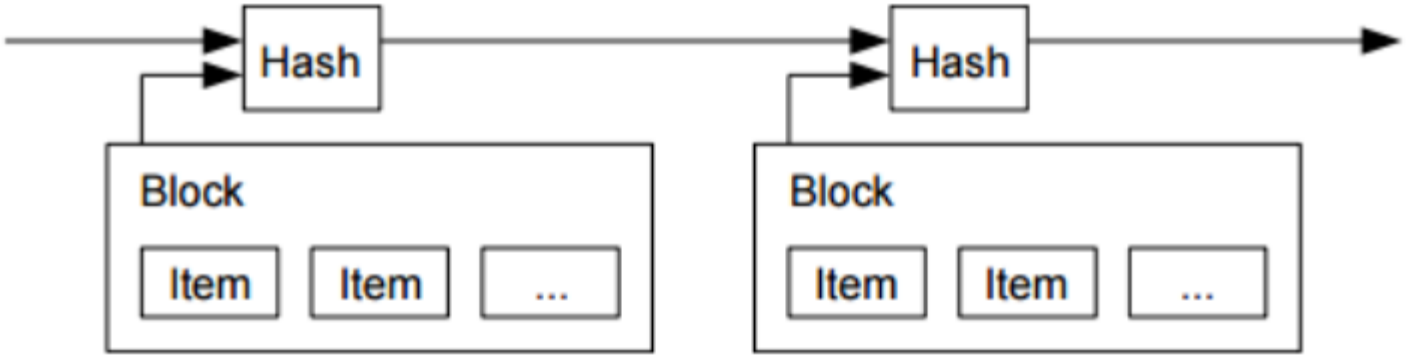
Decentralized Database

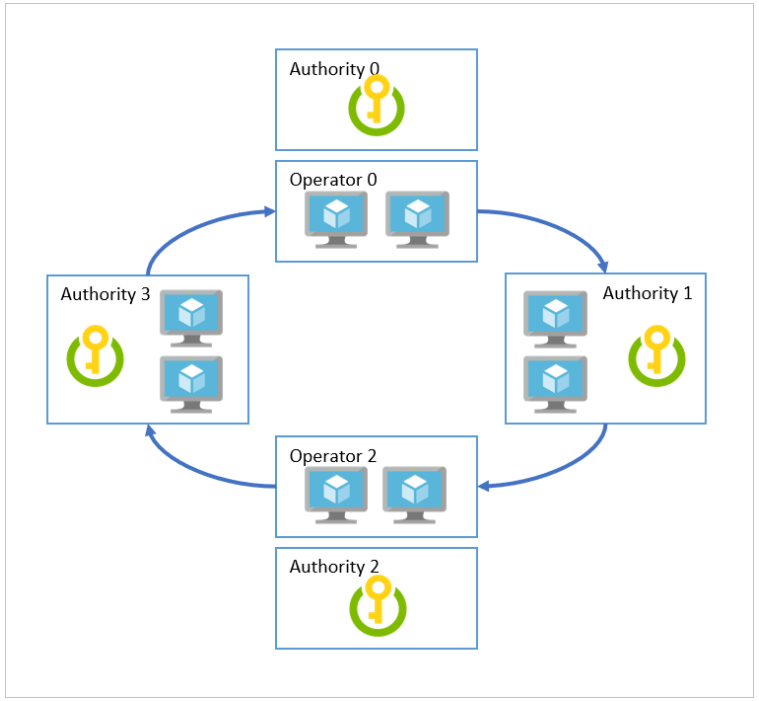
Nodes in the network independently replays transactions to determine the current state of the database

Each account's balance is the result of transactions that reference that account (deposits and debits)

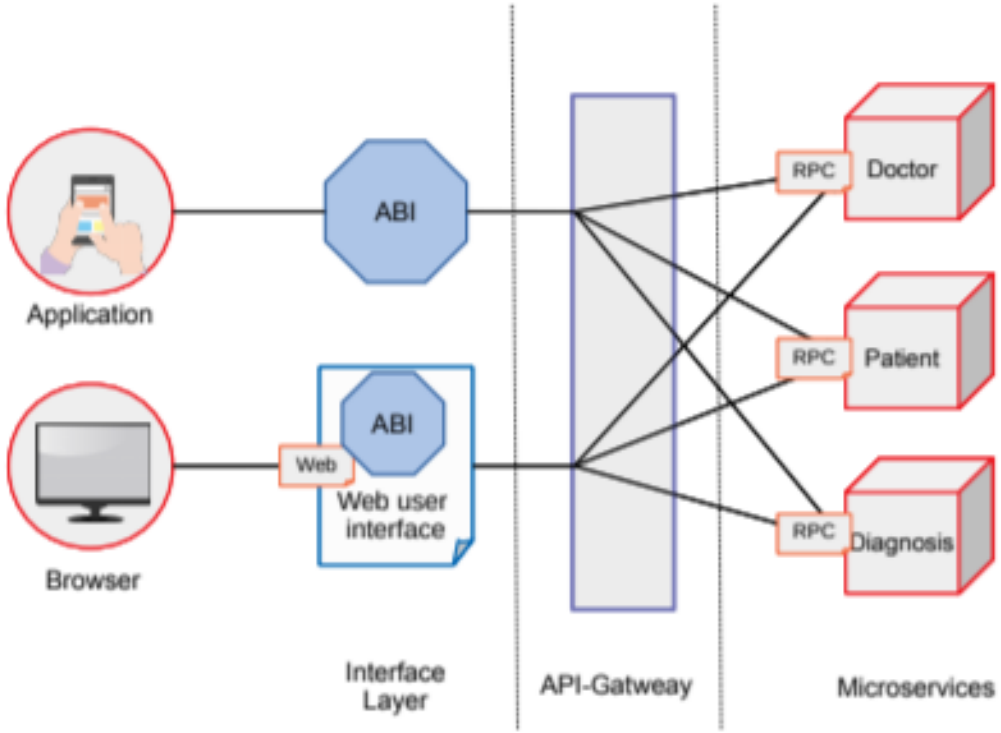
Miners aggregate transactions into blocks

Block hash includes hash of previous block, forming a block chain:

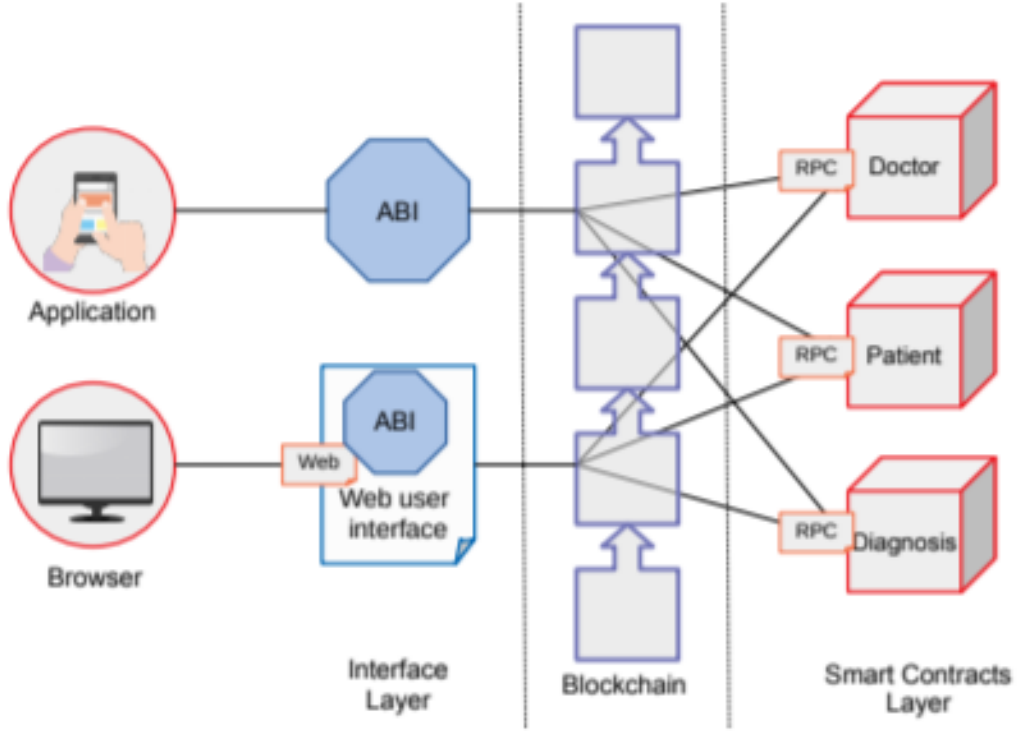




Microservices



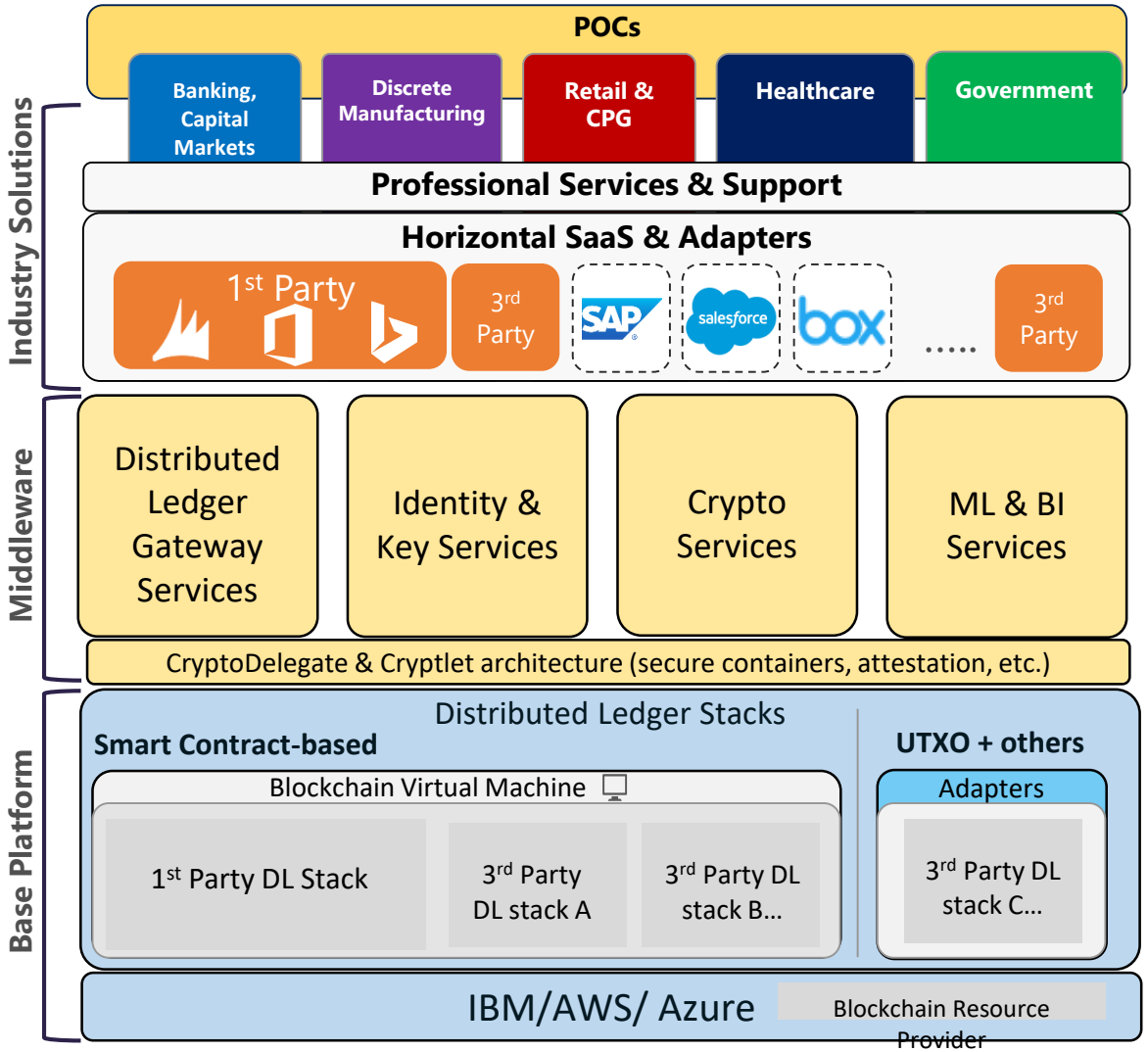
Blockchains



Design Patterns & Practices

Design Principle	Microservice	Smart Contract
Single responsibility	Typically provides a CRUD interface on a single entity.	Defines roles, state and the relevant logic for a validation workflow on a single object, using the CRAB approach (more later in this article).
Context bounded	No dependency on other services, owns its data model for persistent storage.	Does not have dependency on other smart contracts and leverages the on-chain data model (that is, the blockchain itself) as the preferred data model.
Messaging enabled	Can leverage an API gateway for inter-service communication, and a service bus for intra-service communication.	Can leverage “oracles” or “cryptlets” for off-chain data access, or tools like Azure Blockchain Workbench that expose a REST API.
Autonomously developed	Multiple programming languages and frameworks.	Multiple blockchain platforms available, although no cross-platform communication currently exists.
Independently deployable	With proper design (event sourcing, CQRS) can reduce or remove dependency completely.	Similar design patterns apply (described in this article).
Distributed and decentralized	Distributed architecture as opposed to a centralized “monolith.”	Built-in distributed and decentralized digital ledger by design.

Build enterprise-ready blockchain middleware



The Blockchain Middleware will provide core services, which will help users create and build on top of blockchains within Azure

The core services can be broken down into the following:

- **Identity and Certificate Services** – Helps with authentication, authorization, access, and lifecycle management.
- **Encryption Services** – Provides encryption for blockchain transactions and fields
- **Cryptlet Services** – Provides runtime for cryptlets and communication between blockchain and cryptlet trusted host
- **Blockchain Gateway Services** – Provides communication between multiple blockchains
- **Data Services** – Rich data services, such as analytics, auditing, and machine learning
- **Management and Operations** – Tools for deployment and

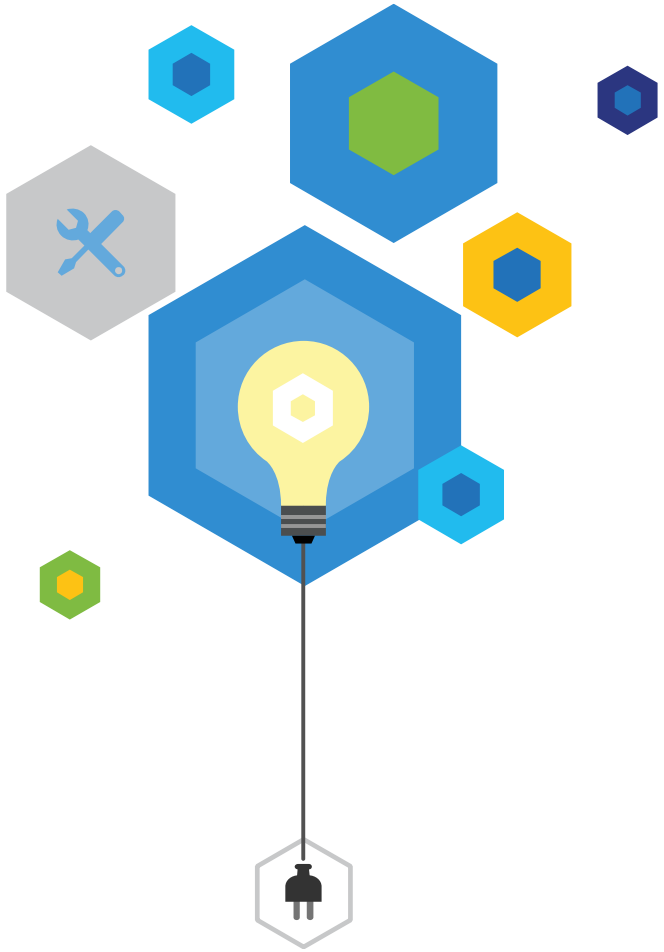
Blockchain development

- Current landscape of tools is highly fragmented
- **Frameworks** – Truffle, Nethereum, Blockapps, Tendermint, Go-Ethereum, Tierion
- **Dev Blockchains** – Strato, Chain, BigChainDB, EthereumJS TestRPC, HyperLedger
- **IDE based** – Mix, Cosmo, Ethereum Studio, Visual Studio, VSCode, Sublime Package, Atom Ethereum Interface/Linter, Emacs Solidity, Vim Solidity

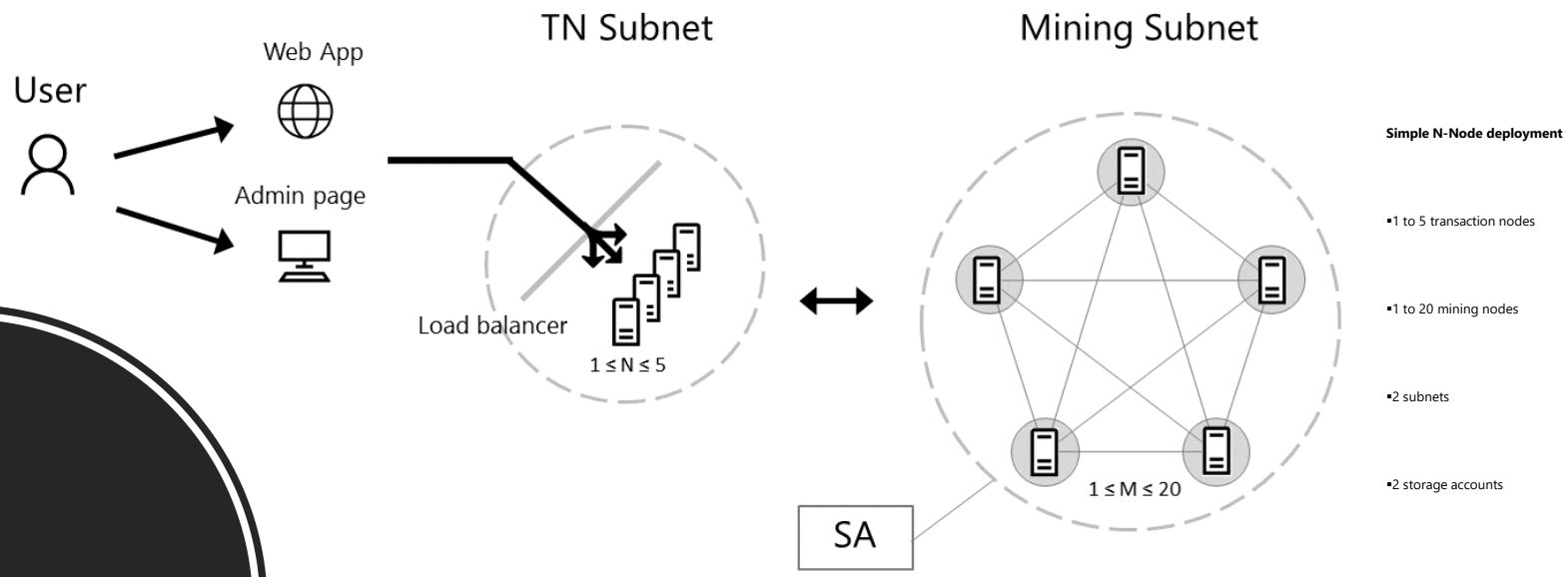
FRAMEWORKS/SDK

IDE INTEGRATION

DEVELOPMENT CHAINS



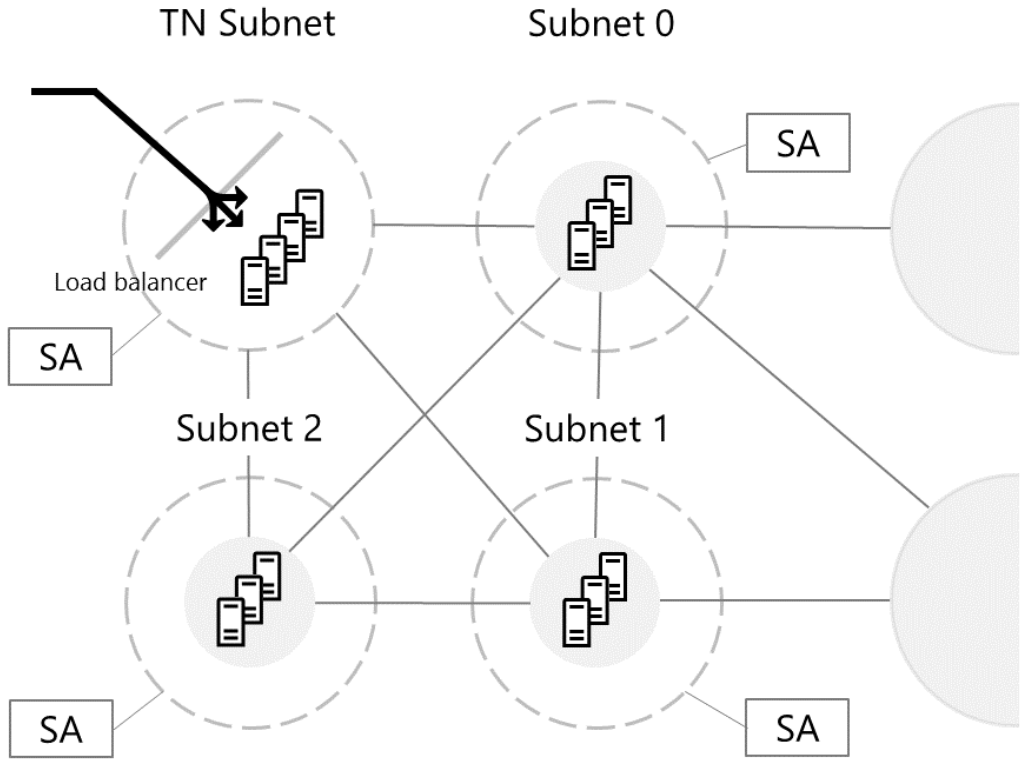
Demo: Provisioning and Developing Blockchain on Azure



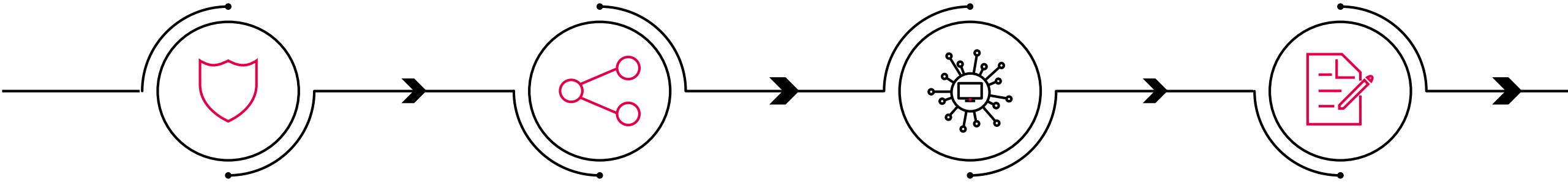
ARM Template

Mock Consortium Network

- 1 to 5 transaction nodes
- 1 to 100 mining nodes
- 2 to 6 subnets
- 2 to 6 storage accounts



Blockchain on Azure



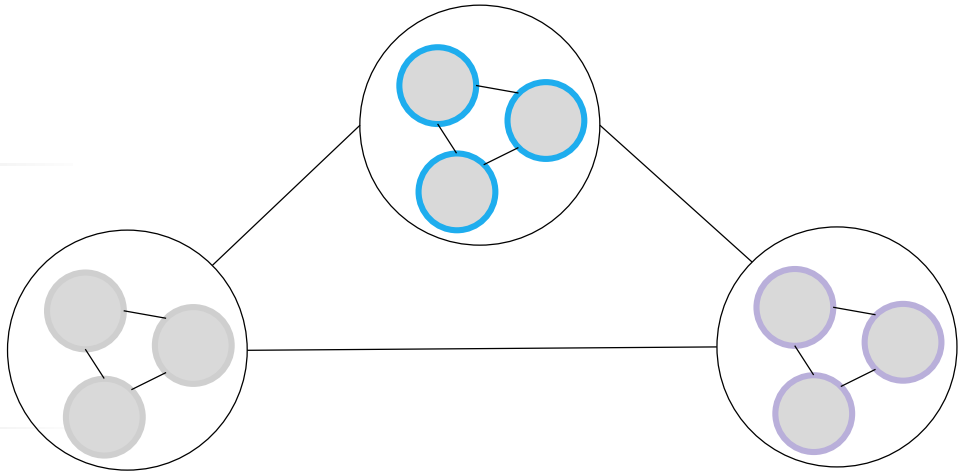
Secure **Shared** **Distributed** **Ledger**

Coco Framework: foundation of blockchain for enterprise

Scalability
Database-like speeds for transaction throughput and latency

Confidentiality
Richer and more flexible confidentiality models

Consortium Governance
Configurable constitution to govern membership



Confidential Computing

Based on Trusted Execution Enclaves (TEEs)

Windows Server Virtual Secure Mode

Intel SGX

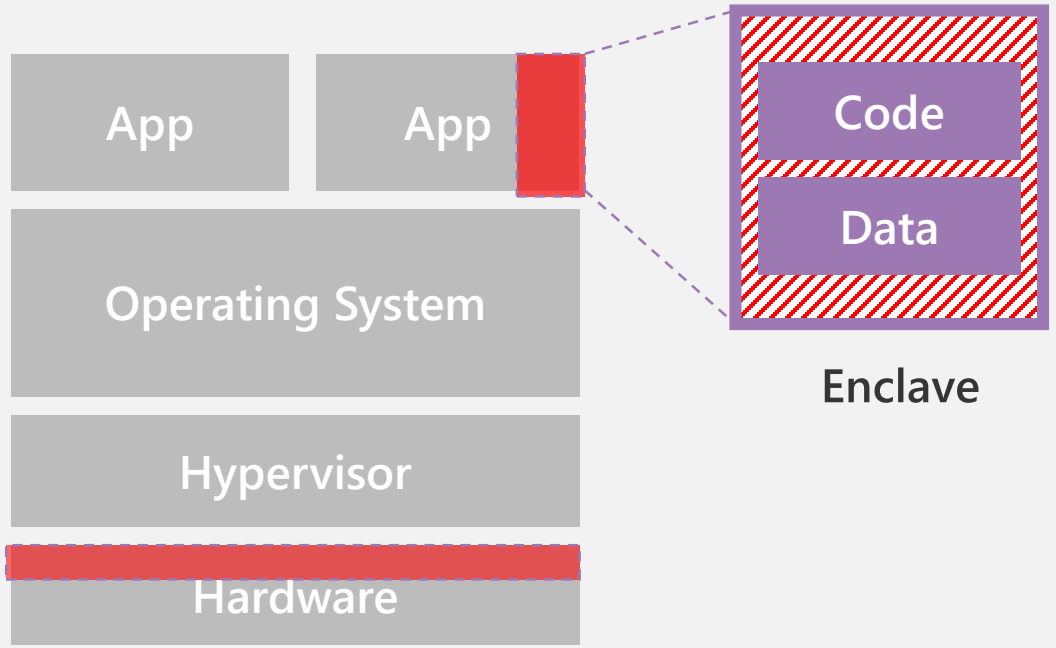
Secures all data while in use

Customer workloads are invisible to host fabric

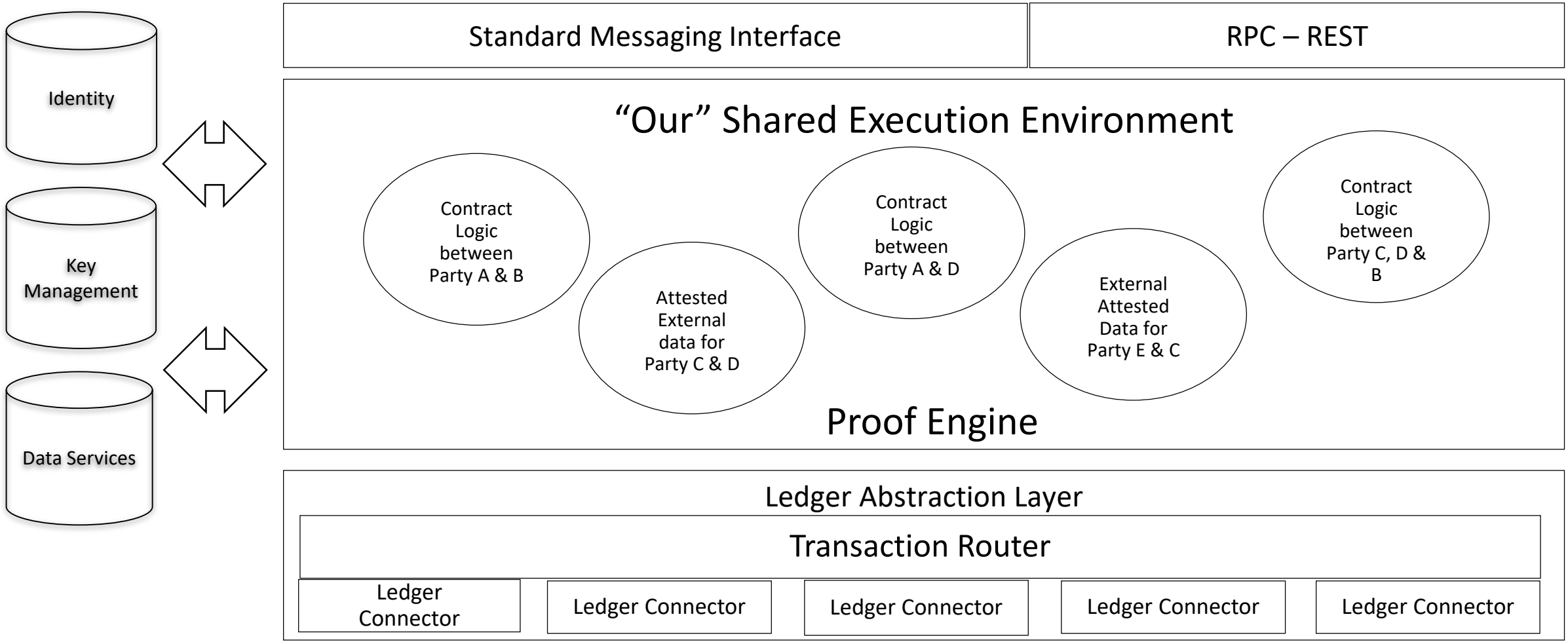
Customer data is always encrypted – during compute and storage

Protects against multiple threats

Malicious insiders

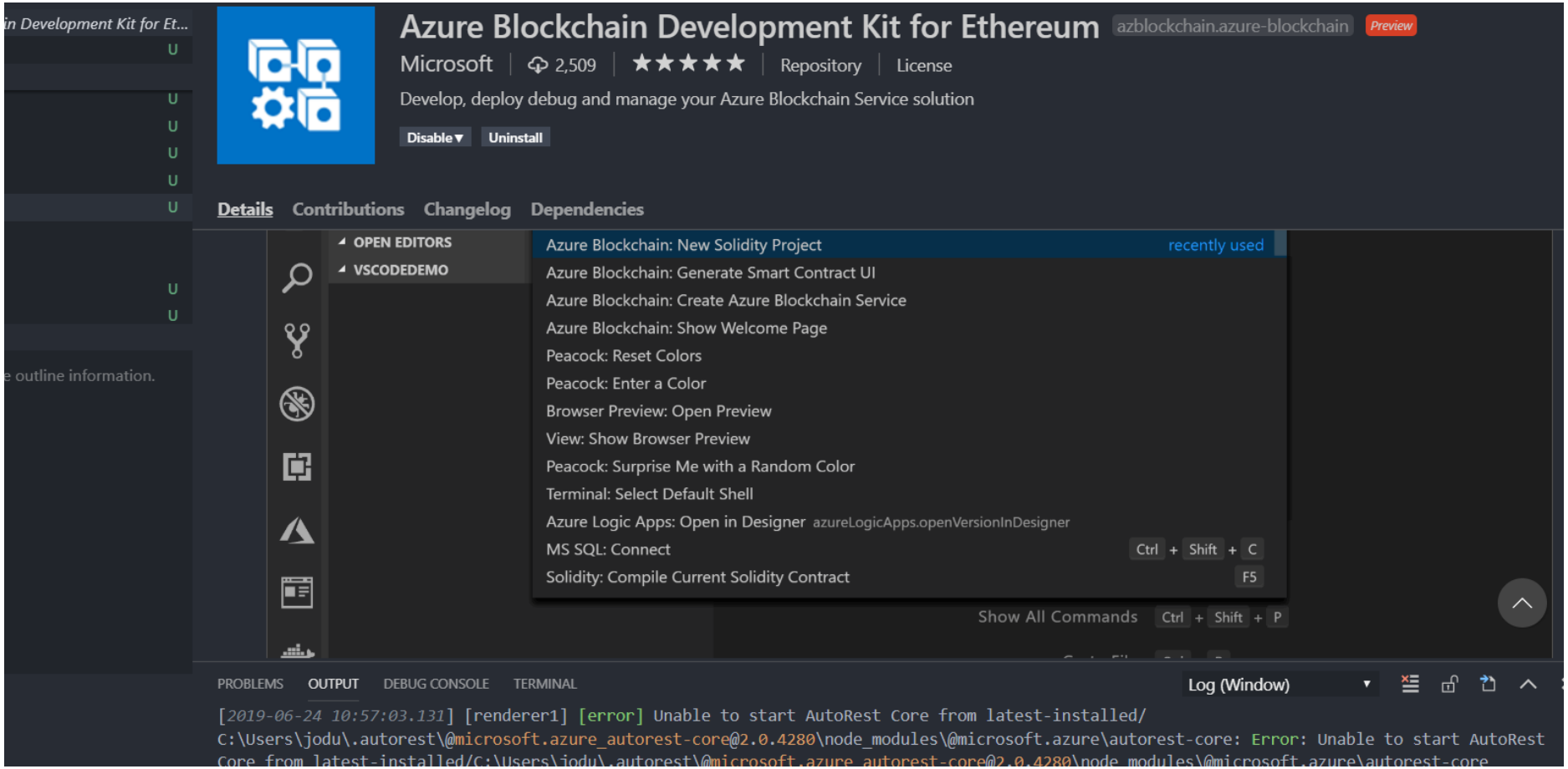


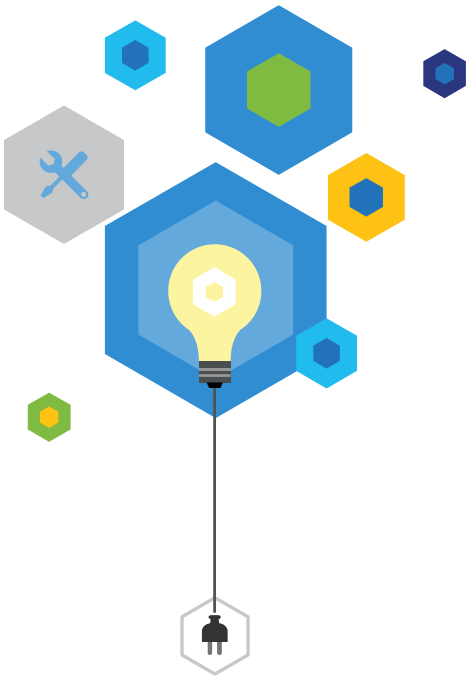
Microsoft DL Layers

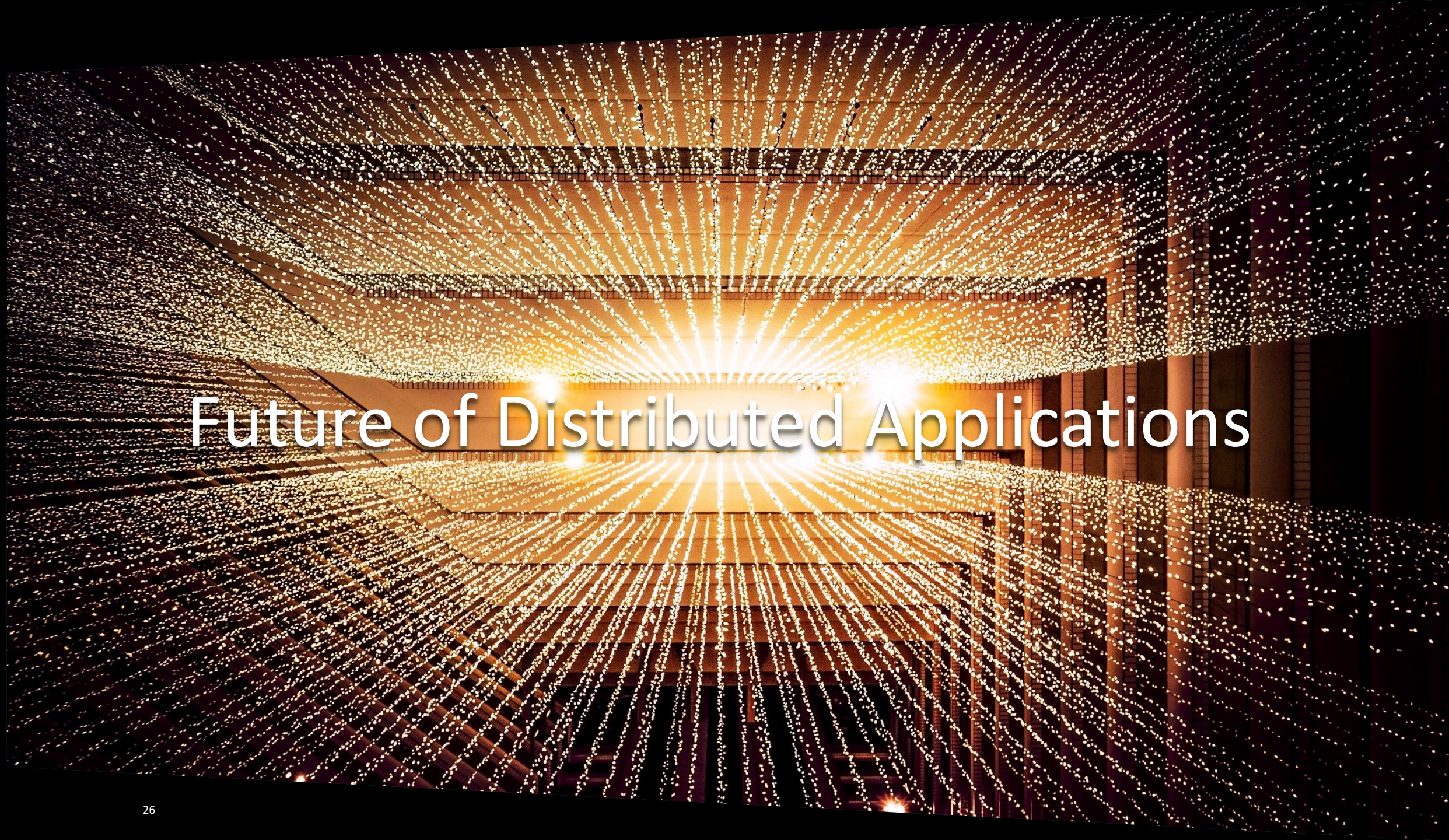


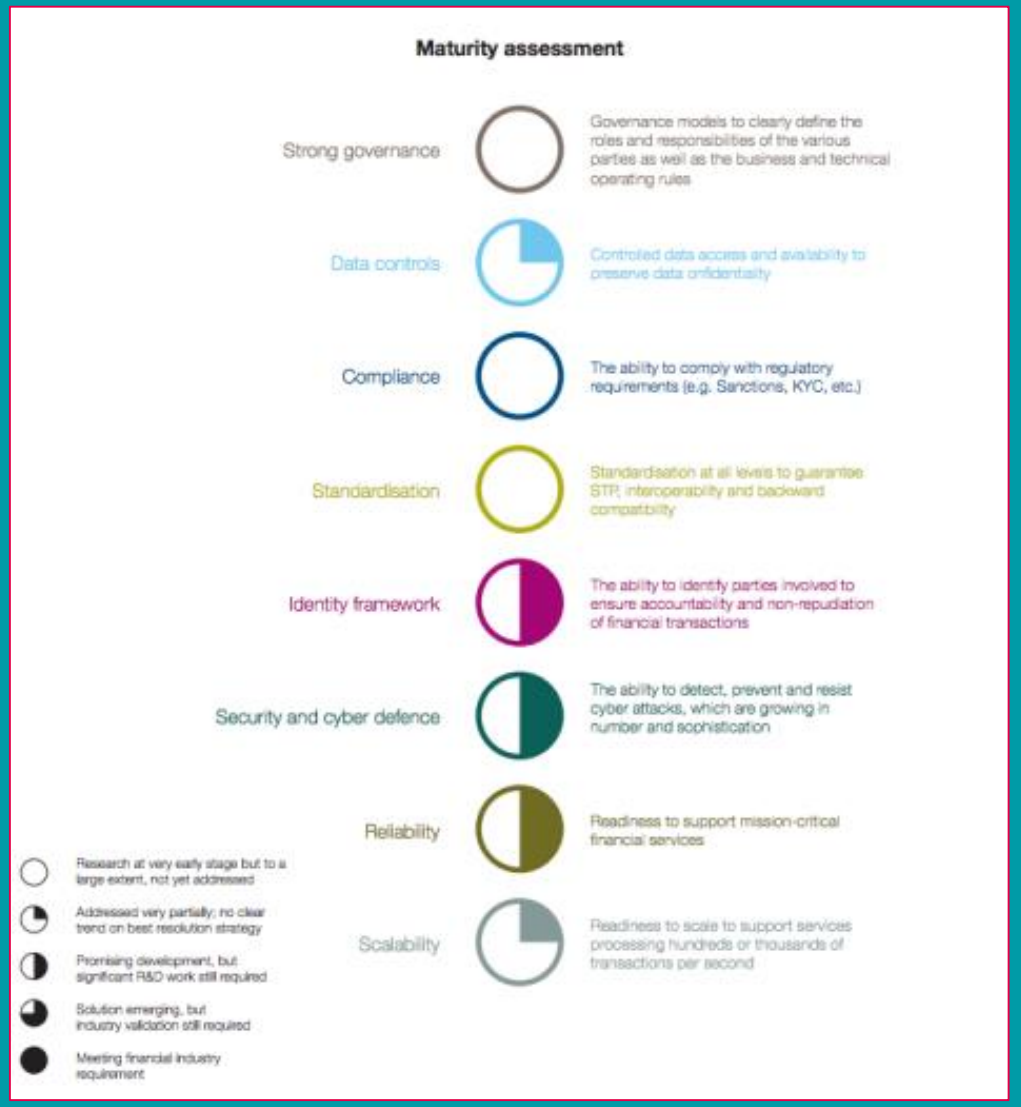
corda ethereum ENTERPRISE ETHEREUM ALLIANCE Chain HYPERLEDGER

Demo: Azure BlockChain Workbench & VSCode Extension






Future of Distributed Applications



A View of Distributed Ledger Technology:

Promising Development in Security, Identity and Reliability

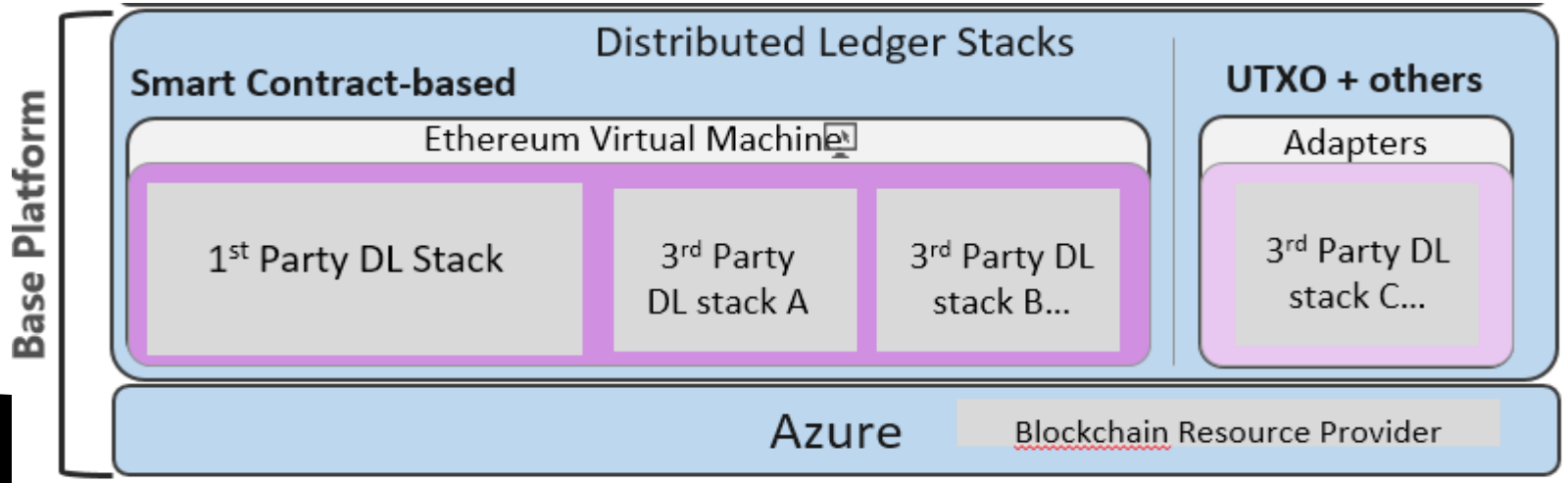
Blockchain as a service

Bletchley

Ethereum Consortium Blockchain Network

3 weeks = multi-party
Private consortium network

8 Questions, 5-8 minutes
=
4 to 100’s of nodes



Blockchain network types

Public

Trustless – POW/Mining

- Bitcoin
- Ethereum
- *n* Crypto-currency networks

Private

Semi-Trusted - *consensus algos

- Ethereum
- Enterprise Ethereum – Quorum
- Hyperledger – Fabric
- Chain
- R3 Corda
- *n* private blockchain platforms

Consensus

Agreement on the order of transactions

Different ordering can lead to double spending

Problem

No one trusts anyone else

Solution

Accept proposals for transaction order and reward winning proposals

Smart Contract Basics

Ethereum VM (EVM) environment:

- 32-byte key/value map durable storage
- 1024x32-byte stack while contract executes
- Infinite 32-byte memory array while contract executes

Smart contract development

- Solidity high-level language, others
- Compiled to EVM bytecode

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

Block #0

Summary	
Number Of Transactions	1
Output Total	50 BTC
Estimated Transaction Volume	0 BTC
Transaction Fees	0 BTC
Height	0 (Main Chain)
Timestamp	2009-01-03 18:15:05
Difficulty	1
Bits	486604799
Size	0.285 KB
Version	1
Nonce	2083236893
Block Reward	50 BTC

Using Transactions & Proof of Work example

Goal

Find a hash that starts with 0000 of "hello, world!" plus a *nonce*

```
"Hello, world!0" => 1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
"Hello, world!1" => e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
"Hello, world!2" => ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7
...
"Hello, world!4248" => 6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc0b965
"Hello, world!4249" => c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6
"Hello, world!4250" => 0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
```

What about some Code !
Smart Contracts vs Data/Service Contracts, Transactions & Consense

Smart Contracts Integration

Smart contracts can only access:

- Their own storage
- Some information about the block chain
- Public smart contract functions

Cannot call Web services, OS functions, etc.

Must be completely deterministic

External information must be injected into the blockchain

- Stored as contract storage
- “Oracles” can expose functions that return the data

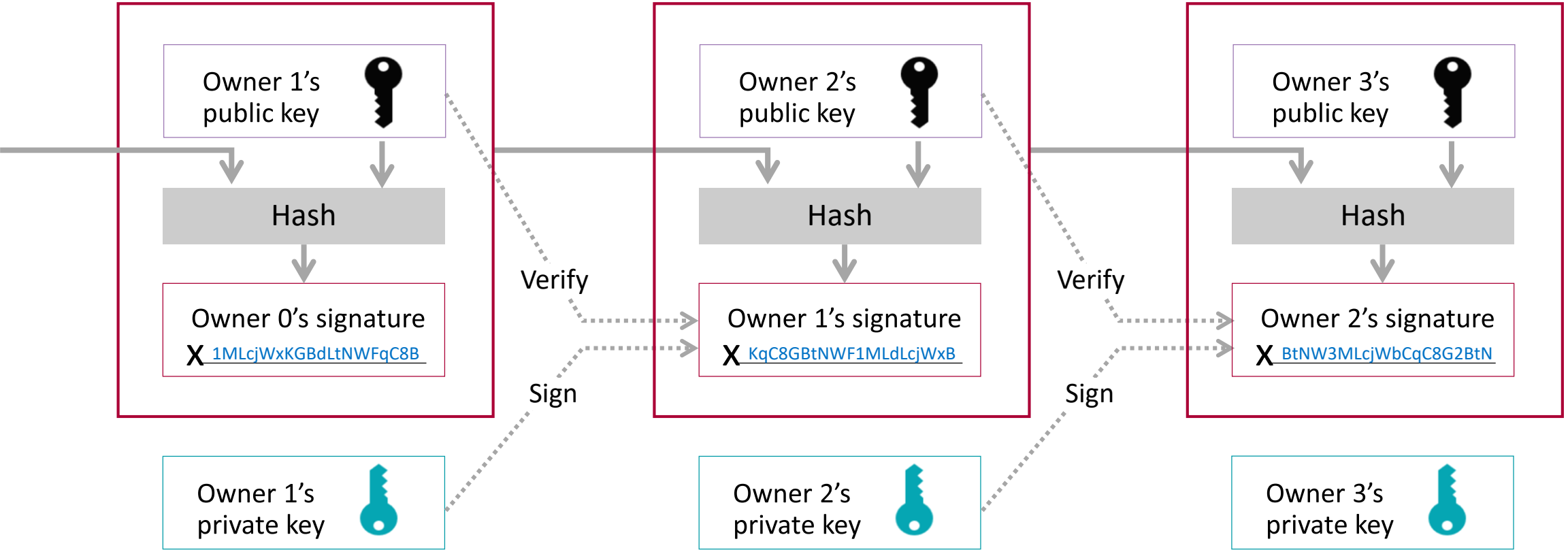
Stateful microservices are reliable and consistent

- Each service is backed by replica set to make its internal state reliable
- All replicas are logically consistent – meaning all replicas see the same linearized order of read and write operations to initial state
- Read-Write quorums are supported and are dynamically adjusted
- Replica set is dynamically reconfigured to account for replica arrivals and departures

Transaction Chains

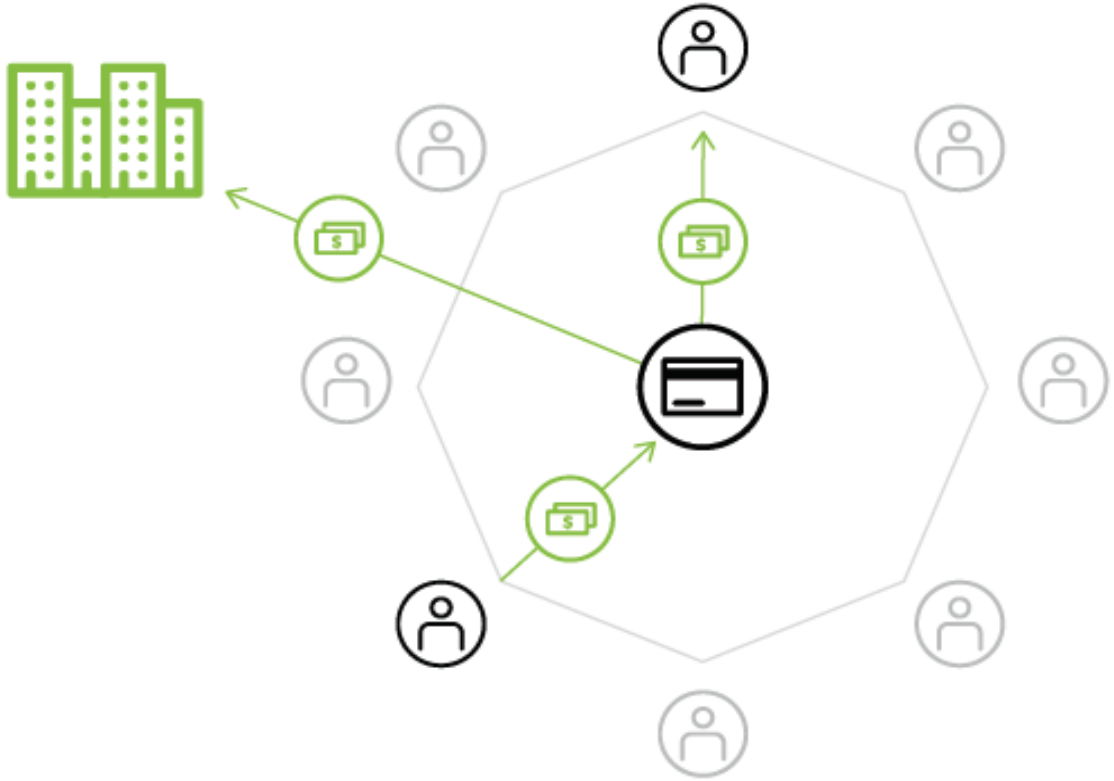
Coin is defined as chain of digital signatures

- Transactions transfer coin from an owner to someone else
- Coins transferred by signing hash of previous transaction and public key of next owner

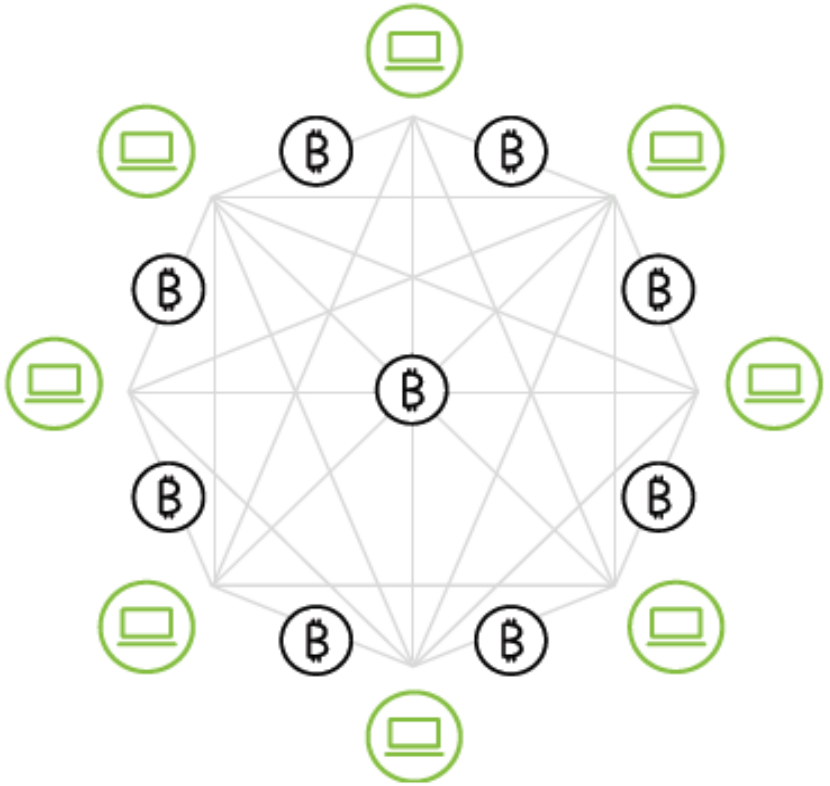


Microservices transactions placement

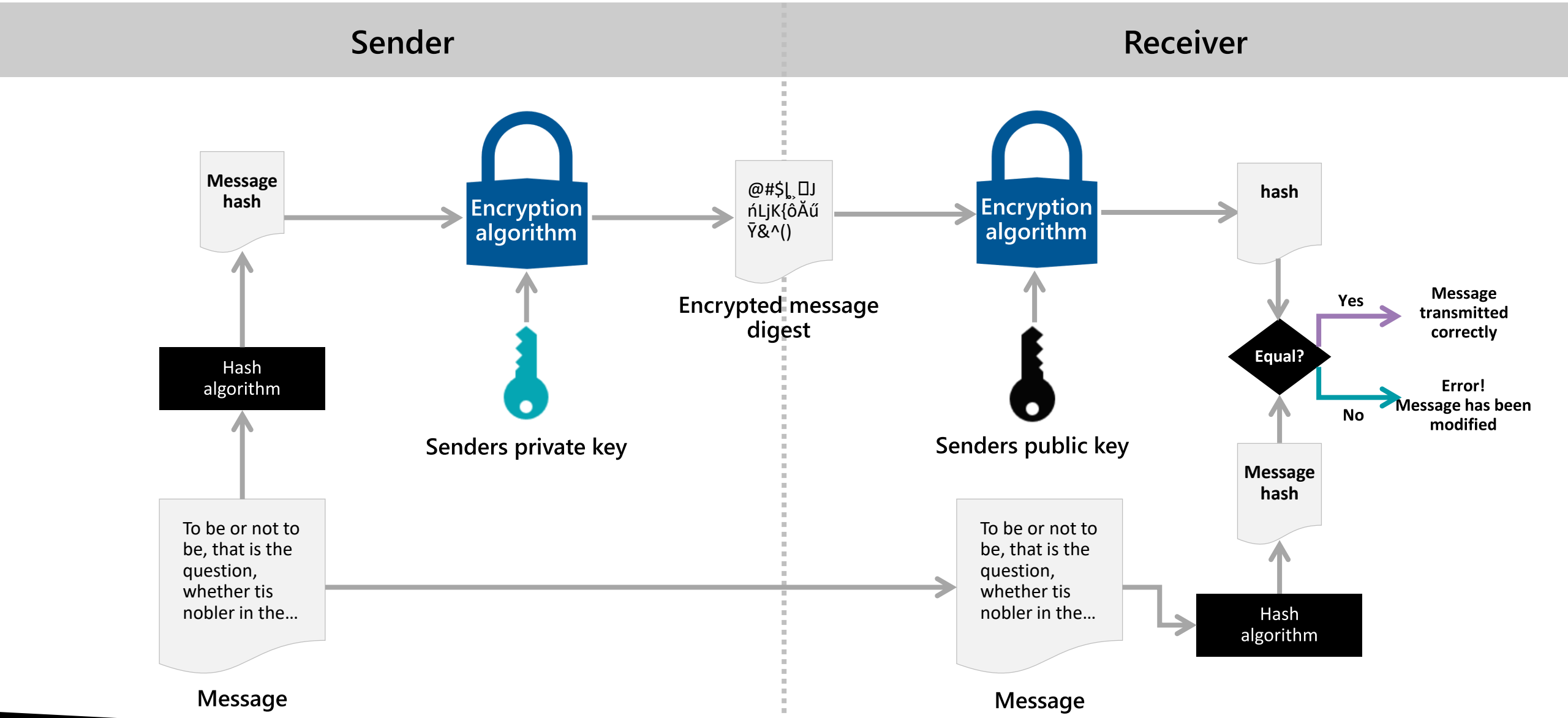
Central



Decentral



Block Encryption Flow



Decentralized Autonomous Organizations (DAOs)

Organization whose bylaws and behavior is entirely code-based

Allows a group of members to vote on proposals

“The DAO”: Ethereum-based system for crowd-funding

- Members contribute Ether
- Vote on projects to fund
- No centralized owner
- No regulatory oversight

```
function executeProposal(uint _proposalID) returns (int result) {
    Proposal p = proposals[_proposalID];
    /* Check if debating period is over */
    if (now > (p.creationDate + debatingPeriod) && p.active){
        uint quorum = 0;
        /* tally the votes */
        for (uint i = 0; i <  p.votes.length; ++i) {
            Vote v = p.votes[i];
            uint voteWeight = voterShare.coinBalanceOf(v.voter);
            quorum += voteWeight;
            result += int(voteWeight) * v.position;
        }
        /* execute result */
        if (quorum > minimumQuorum && result > 0 ) {
            p.recipient.call.value(p.amount)(p.data);
            p.active = false;
        } else if (quorum > minimumQuorum && result < 0) {
            p.active = false;
        }
        ProposalTallied(_proposalID, result, quorum, p.active);
    }
}
```

Smart Contract Greeter

```
contract mortal {
    /* Define variable owner of the type address*/
    address owner;

    /* this function is executed at initialization and sets the owner of the contract */
    function mortal() { owner = msg.sender; }

    /* Function to recover the funds on the contract */
    function kill() { if (msg.sender == owner) selfdestruct(owner); }
}

contract greeter is mortal {
    /* define variable greeting of the type string */
    string greeting;

    /* this runs when the contract is executed */
    function greeter(string _greeting) public {
        greeting = _greeting;
    }

    /* main function */
    function greet() constant returns (string) {
        return greeting;
    }
}
```

https://github.com/ethereum/go-ethereum/wiki/Contract-Tutorial

Microservice Actor Events

- Used for call-backs to clients that are not actors

  Do not use between actors

```
public interface IPlayerEvents : IActorEvents
{
    void GameStarted(string otherPlayer, string yourRole);
    void GameStateChanged(string cells, PlayerGameStatus
status);
}
```

```
public interface IPlayer : IActor,
IActorEventPublisher<IPlayerEvents>
```

```
var evt = GetEvent<IPlayerEvents>();
evt.GameStateChanged(cells, status);
```

```
public class MainViewModel : ViewModelBase, IPlayerEvents
```

Microservice Actor State

- Simply a class that supports DataContract serializer

  External from the class that implements the actor

```
[Readonly]
public async Task SelectCell(int cellId)
{
    var game = Game.FromId(State.CurrentGameId);
    await game.MakeMove(cellId, this);
}
```

```
[DataContract]
public class GameManagerData
{
    [DataMember]
    public string Initiator;
}
```

Volatile Actor State Provider

Keeps the state only in memory but replicated over a number of servers

Def per actor or per assembly

Default: On disk using a key-value store

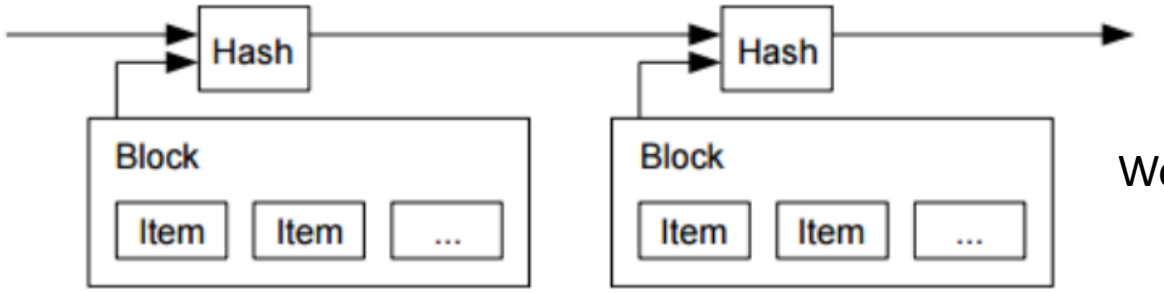
```
public class GameManager :
  Actor<GameManagerData>, IGameManager
{
    public async Task LetMePlay(IPlayer player)
    {
        var playerId = Player.GetId(player);

        if (State.Initiator == null)
        {
            State.Initiator = playerId;
```

The Transaction Order Problem

Challenge is agreeing on transaction order to prevent double-spend

Ideal system uses a global timestamp to determine transaction order

- Each timestamp includes the previous in its hash, forming a chain
- Problem: no trusted authority to serve as timestamper

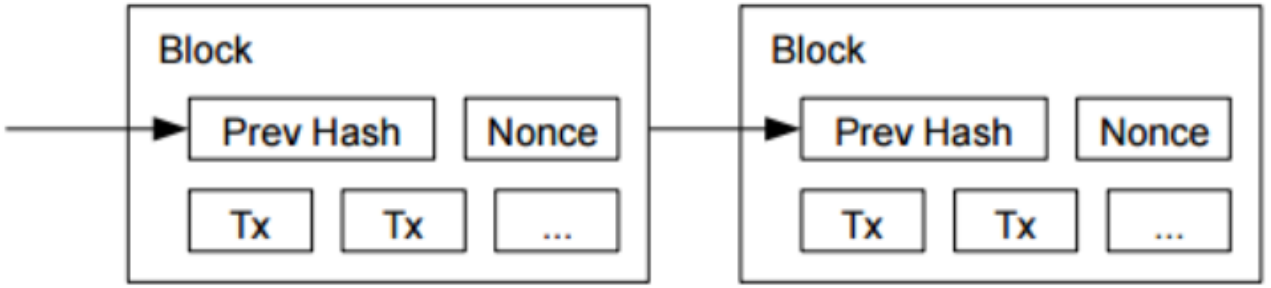


Work is finding a hash of block that has a required number of 0's

Miners increment a nonce until they solve the puzzle

Hash includes previous block hash

As network becomes more powerful, work is adjusted to be more difficult



Smart contract Example Vulnerability

Cross function race condition:

Can be called by attacker
to transfer unlimited amounts

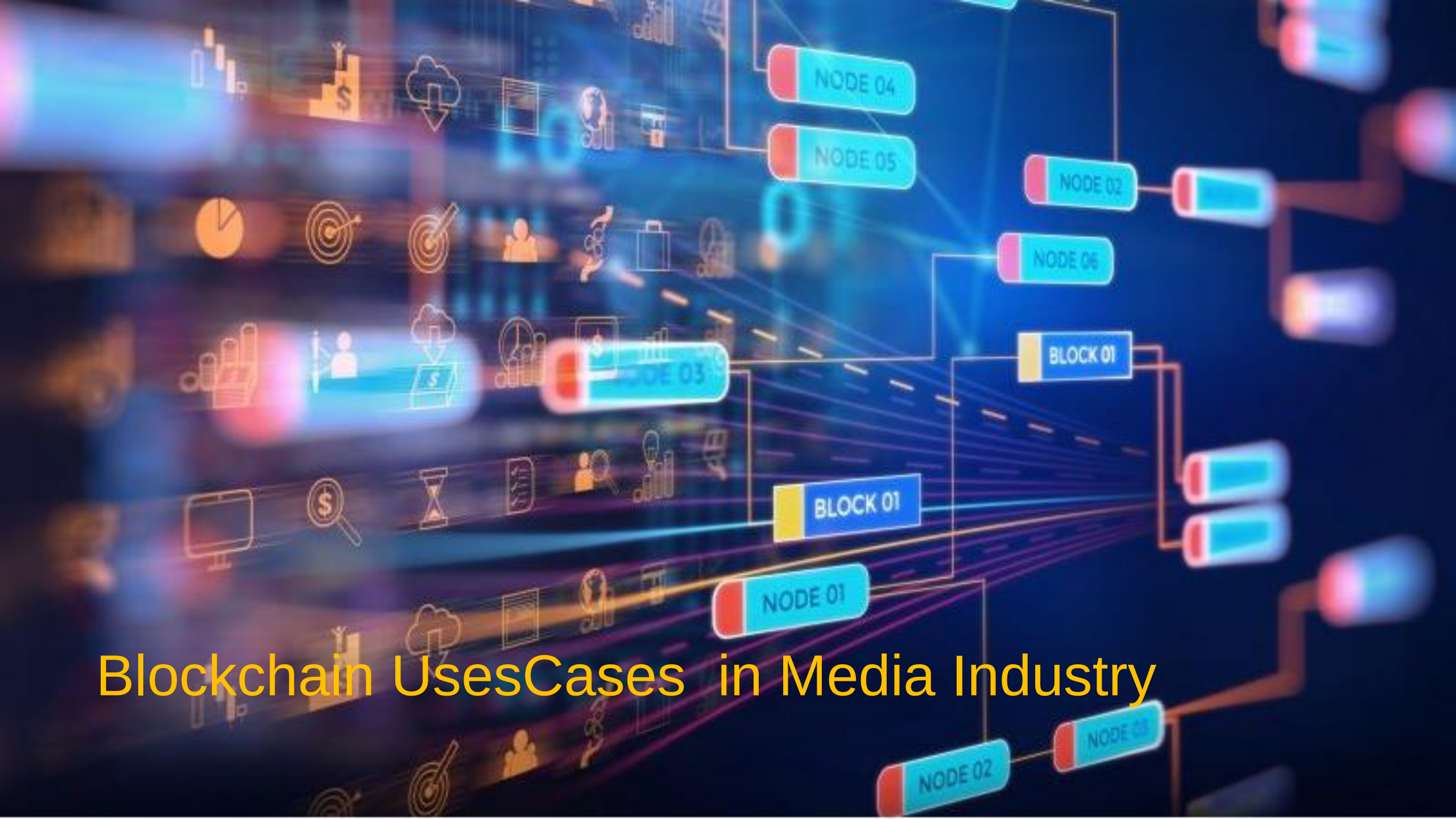
```
// INSECURE
mapping (address => uint) private userBalances;

function transfer(address to, uint amount) {
    if (userBalances[msg.sender] >= amount) {
       userBalances[to] += amount;
       userBalances[msg.sender] -= amount;
    }
}

function withdrawBalance() public {
    uint amountToWithdraw = userBalances[msg.sender];
    if (!(msg.sender.call.value(amountToWithdraw)())) { throw; }
    userBalances[msg.sender] = 0;
}
```

External Call

Similar vulnerability by DAO ha

Balance updated *after* call

Demo:
Provisioning and Developir



Blockchain UsesCases in Media Industry


Conclusion

TAKE AWAY : Blockchains/Ledgers and Microservices

- A Microservice Designed Application or platform can easily integrate or use Blockchain technology
- Blockchain technology is the hype-word de jour, but it's not quite ready for retail transactions—such as payments (you need more stuff…)
- Blockchain in its simplest form is a distributed, shared database technology that works like a public ledger (see next slide)
- When distributed among known and vetted participants, public ledgers appear to work best in complex and unexciting processing environments (they're not sexy) e.g., Smart contracts, custody/settlement, insurance
- They can be very efficient and save a ton, but there are few standards for building and deploying them so far….
- Bitcoin and other virtual currencies per se do not yet offer adequate protection for financial services (and do little for related business automation so far)

What’s next / what’s missing?

- Lack of good real time debuggers, with watchers and visualizers
- More templates for examples of true decentralized apps (e.g. Blockchain Identity integration, dev ops maturity, unit testing)
- Better build tools for complex contracts including nested deployments
- Library or packages concept for smart contracts in the blockchain
- Smart contract migrations for “updates”
- Reference architectures for various scenarios

Q&A

How to get Started

Bitcoin.org

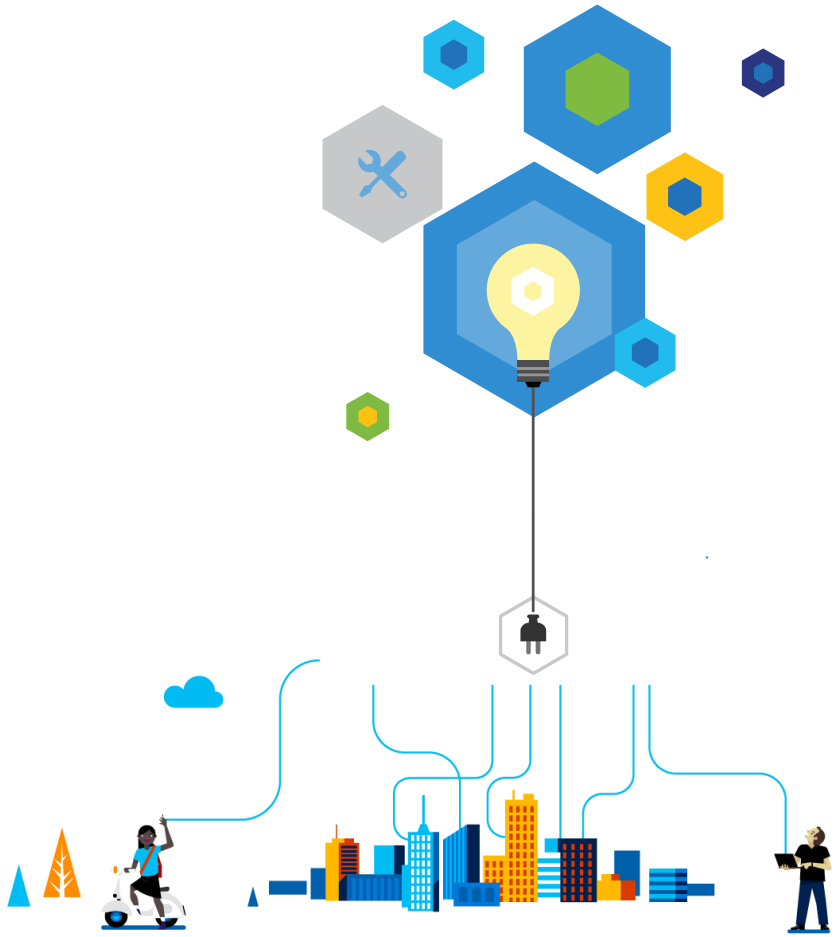
Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto, November 2008

Mastering Bitcoin: Unlocking Digital Cryptocurrencies, by Andreas M. Antonopoulos, O'Reilly Media, December 14

Ethereum White Paper

Blockchain Platform:

- AWS Hyperledger template
- Azure Hyperledger Fabric
- IBM – Hyperldger Fabric



Get Started

Deploy Dev/Test environment(link)
Azure Quick Start Template (Consortium Network)
Ethereumjs TestRPC for quick testing of smart contracts (link)
Truffle – Development framework
Ethereum Studio – Full IDE in the browser
Visual Studio Solidity extensions – VS VSCode

Vielen Dank!