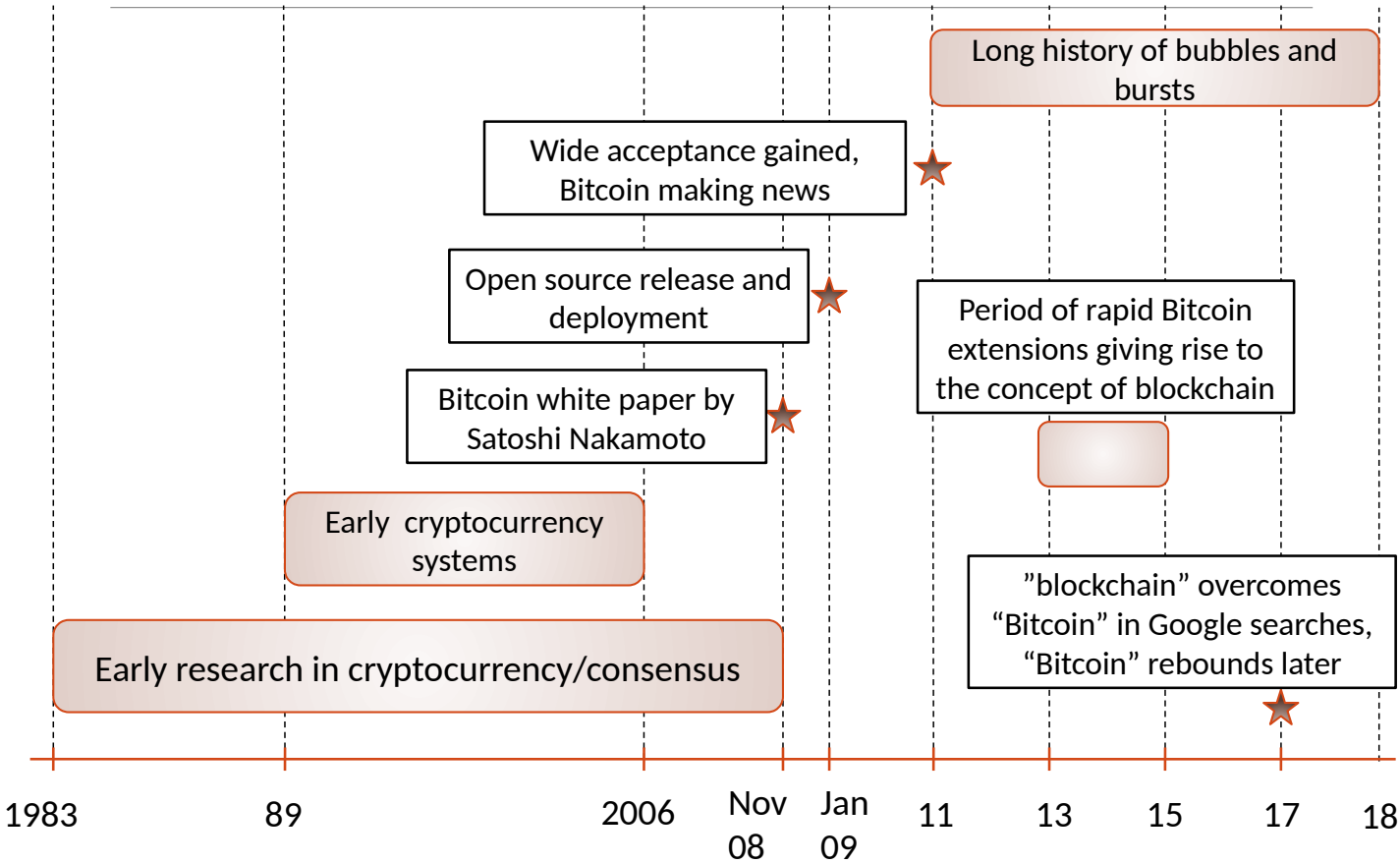
Deconstructing Blockchains: Concepts, Systems, and Insights by Roman Vitenberg

Link to companion papers:
http://msrg.org/papers/bcbi-tr
http://heim.ifi.uio.no/~romanvi/debunking-bc-myths.html

JOINT WORK WITH
KAIWEN ZHANG,
HANS-ARNO JACOBSEN

Historical perspective



Status today: the Blockchain hype

Bitcoin gold rush

40+ percent of top global banks have rolled out full-scale commercial blockchain products so far

- Goldman Sachs alone investing half a billion USD

Blockchain became national storage technology in Estonia

Blockchain storage strategy and regulations in Netherlands

Microsoft declares “blockchain” as a “must win” technology for the Azure platform and business

IBM unveils new blockchain-oriented strategy; opens a new department

Dedicated labs and education programs in blockchain engineering around the globe

- A master program in blockchain engineering at the University of Delft
- A new course at the University of Oslo, TUM, Cornell, and many others

Hottest topic at many societal, industrial, and academic conferences

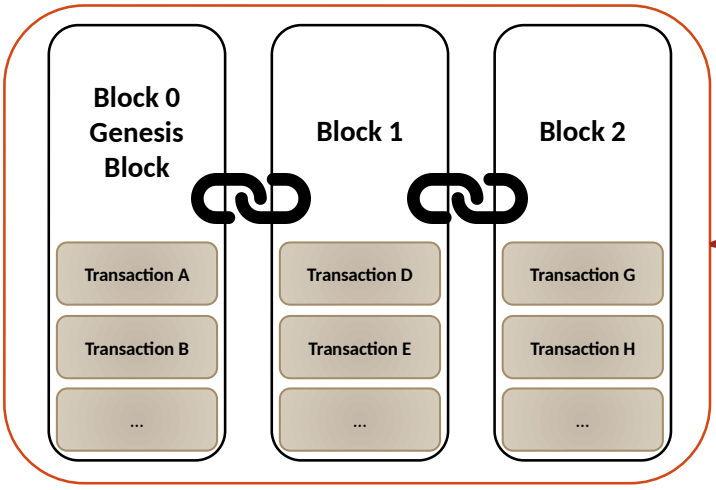
Blockchain 101



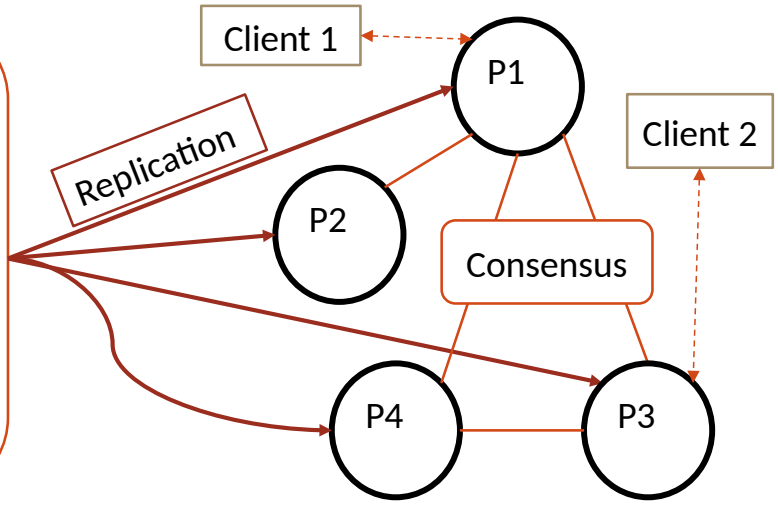
BLOCKCHAIN

Distributed Ledger Technology (DLT)

Blockchain data structure (replicated at every node)



Distributed network



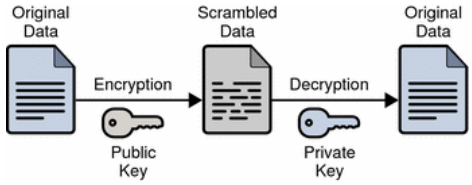
Cryptography is used to…

…encrypt data, prevent modification, insert new blocks, execute transactions, and query…

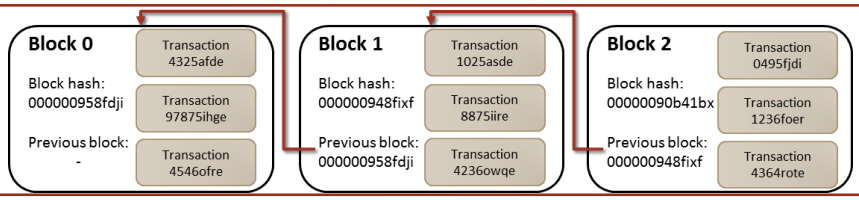
the distributed ledger

Cryptography & Security

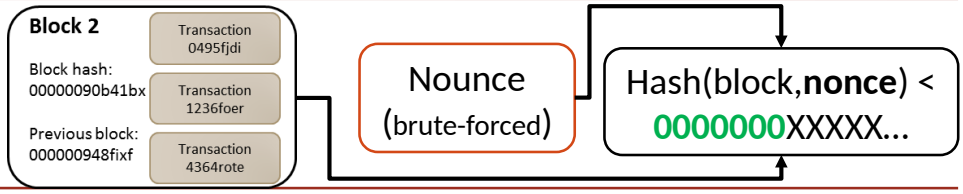
Encrypt data:
Public Key Encryption



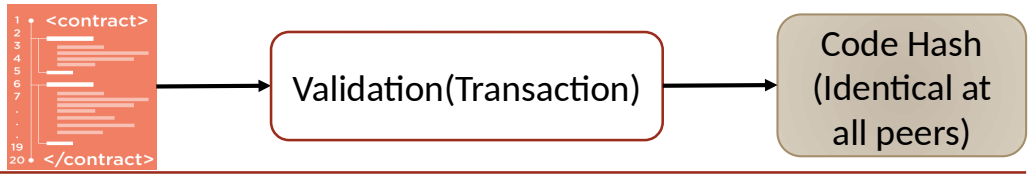
Prevent modification:
Hashed Linked List



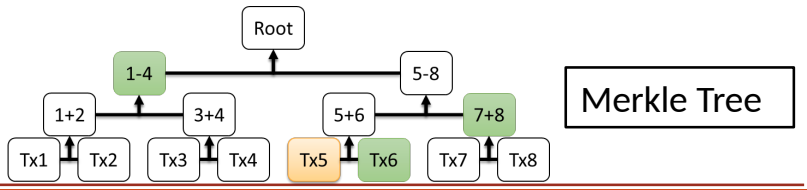
Insert new blocks:
Proof-of-Work



Execute transactions:
Smart Contracts



Query the blockchain:
Simple Payment Verification

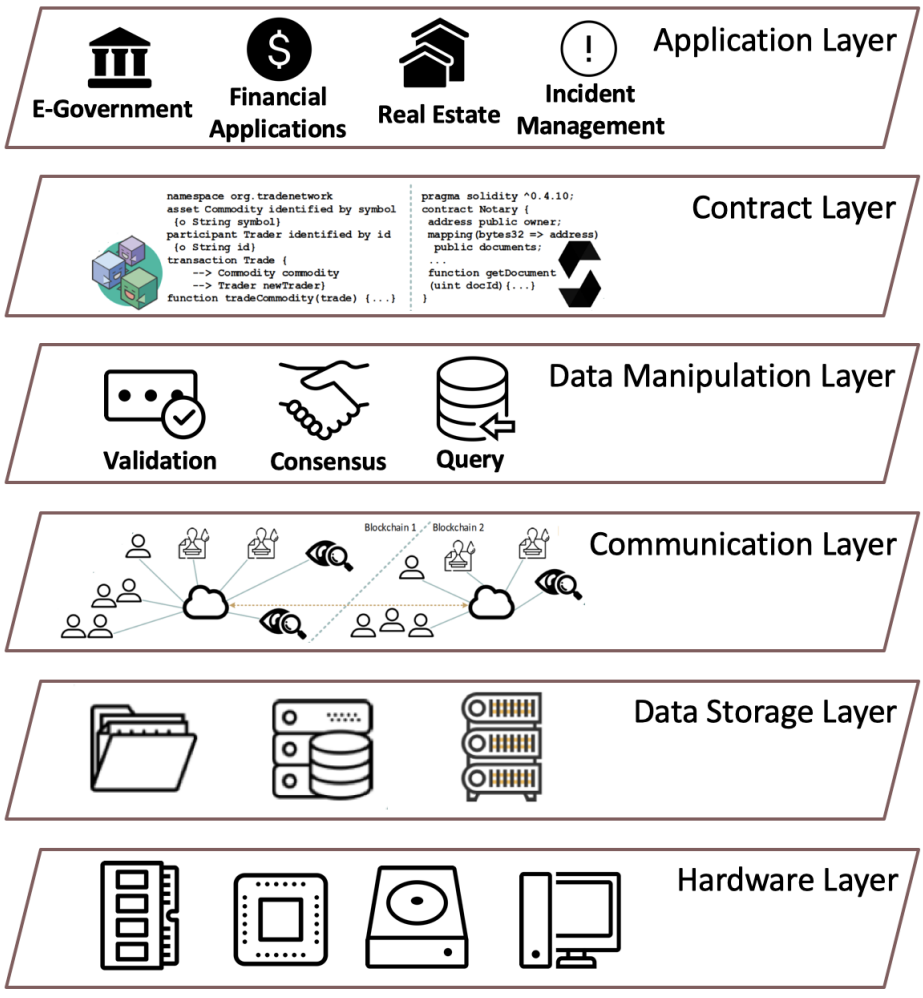


Blockchain Reference Architecture

This vision diagram encompasses all aspects related to blockchain technologies.

Upper layers capture application semantics and their implementation.

Lower layers are concerned with technical system details.



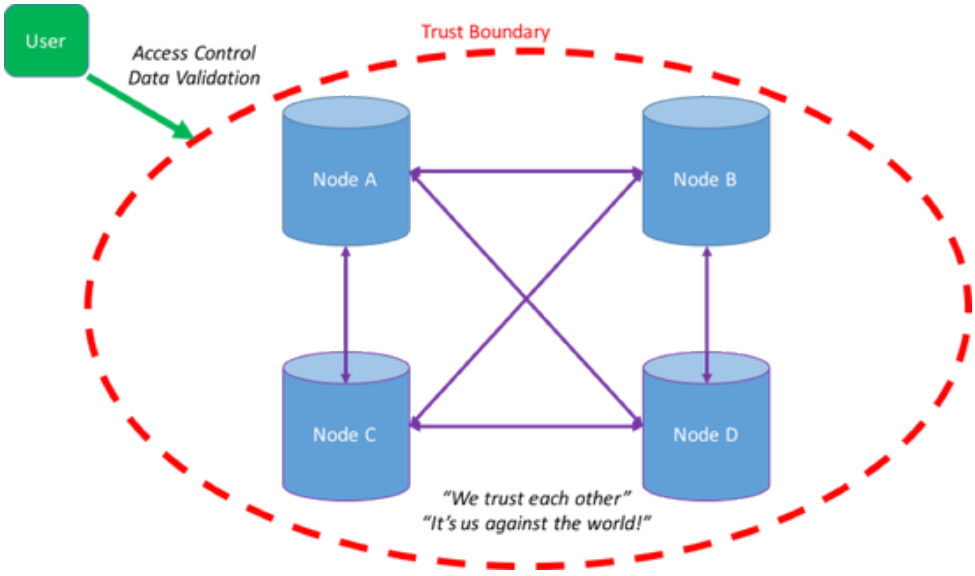
Blockchain vs. Distributed DB

Blockchains maintain a log (aka a ledger) of all transactions since the start of deployment

- e.g. in Bitcoin, there is no direct record of the current state

The trust model is fundamentally different

Distributed database



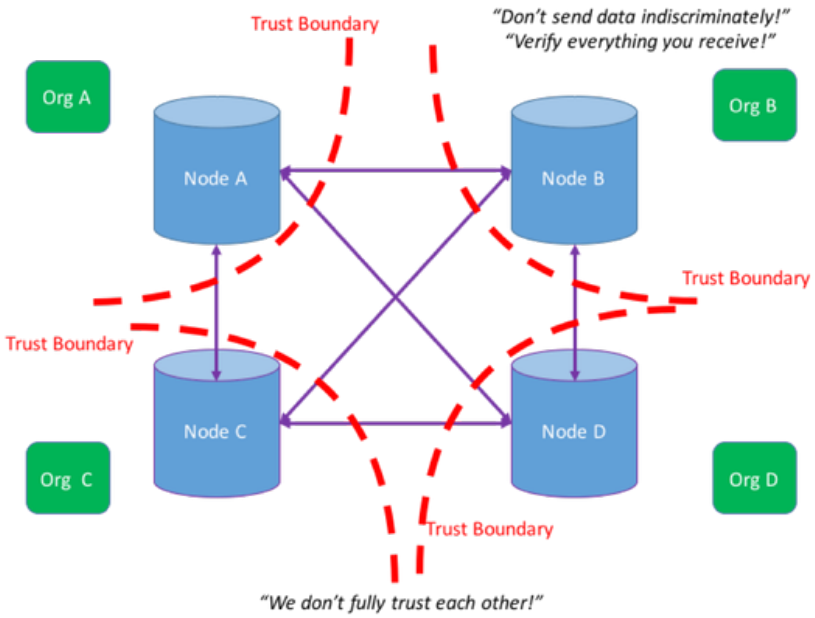
Blockchain vs. Distributed DB

Blockchains maintain a log (aka a ledger) of all transactions since the start of deployment

- e.g. in Bitcoin, there is no direct record of the current state

The trust model is fundamentally different

Blockchain / distributed ledger



Outline

Session 1: Foundations

- Concepts: Byzantine Consensus, Mining, Proof-of-Work, Smart Contracts
- Original system: Bitcoin

Session 2: Beyond Bitcoin

- Smart contracts
- Platforms: Ethereum, Hyperledger

Session 3: Research

- System insights



Blockchain Concepts

DEFINITIONS

BITCOIN OVERVIEW

Bitcoin vs. Blockchain

Bitcoin is a specific system

- Design
- Open-source implementation
- Deployment
- There are alternative cryptocurrency systems (some of which are spawn-offs) but they are not Bitcoin

Blockchain is ambiguous: can be the data structure used in Bitcoin or a separate concept

A guiding design principle/paradigm

- Not even a standard
- Generalization of Bitcoin (In what direction?)
- Hundreds of implementations
- Ethereum alone has hundreds of proprietary deployments in addition to the main public deployment

What is a blockchain-based distributed ledger?

- ✓ *An append-only log* storing transactions
- ✓ Comprised of *immutable* blocks of data
- ✓ *Deterministically* verifiable (using the *blockchain* data structure)
- ✓ Able to execute transactions *programmatically* (e.g., Bitcoin transactions and smart contracts)
- ✓ *Fully replicated* across a large number of peers (called miners in Bitcoin)
- ✓ *A priori decentralized*, does not rely on a third party for trust

Blockchain and the land of ambiguities

Definition 1: a system that uses the blockchain structure of Bitcoin but extends the functionality

- Extended business logic
- Different consensus protocol

Definition 2: a system that maintains a chain of blocks

- Could be a structure other than that of Bitcoin
- Example: Ethereum

Definition 3: a system that maintains a ledger with all transactions

- Not necessarily stored as a chain of blocks
- Example: IOTA

Definition 4: a system with distributed non-trusting parties collaborating without a trusted intermediary

- Example: CORDA

Definition 5: a system that uses smart contracts

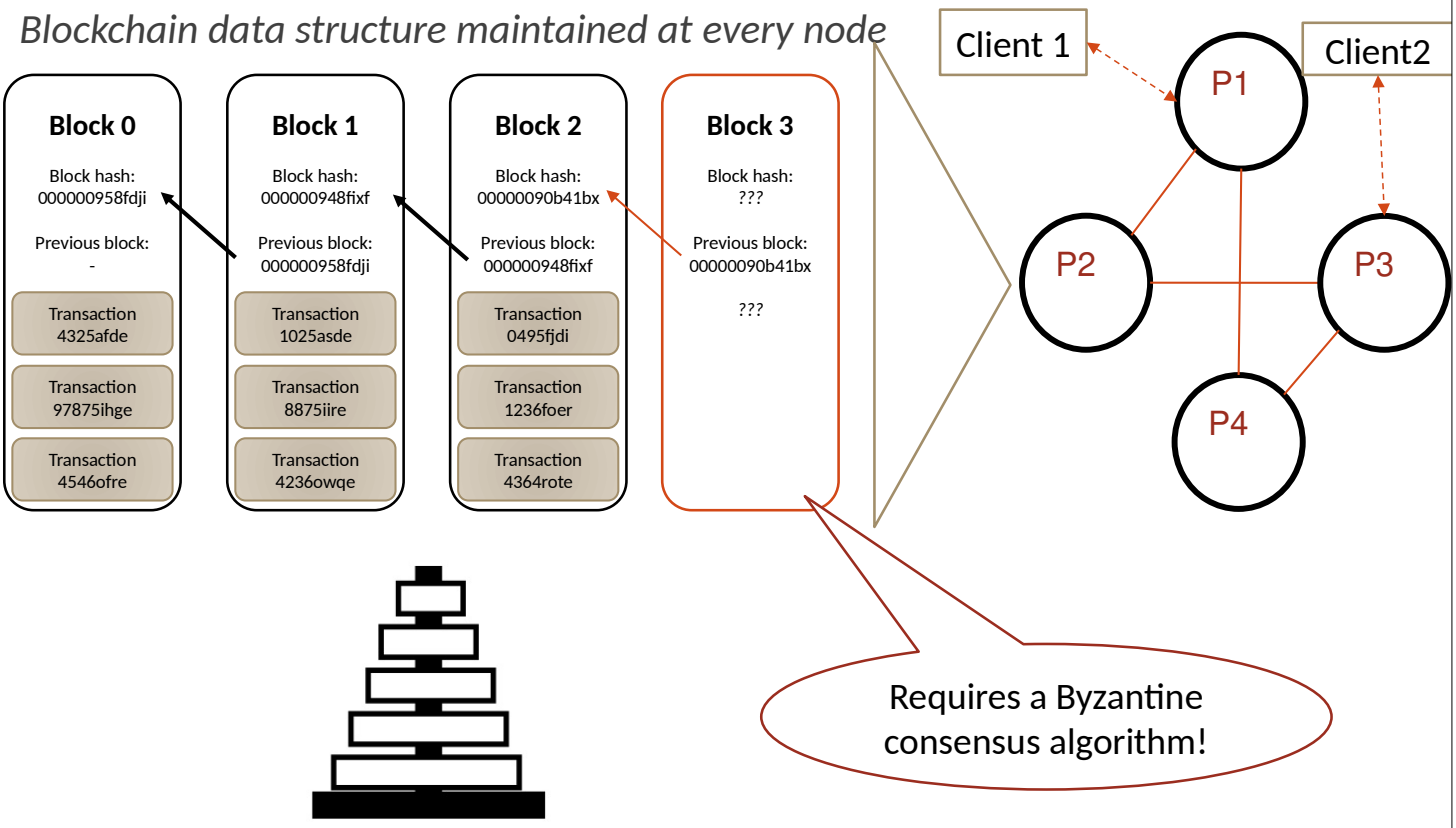
Main benefits of DLTs

Enable parties who do not fully trust each other to form and maintain consensus about the existence, status and evolution of a set of shared facts

The ecosystem of smart contracts

Immutability using Hashing

Blockchain data structure maintained at every node



Blockchain "Cryptopuzzl



verify(nonce, data) meets some "requirements"

Use of "trapdoor functions" (hash functions)

- Cannot reverse the function to find the input
- Therefore, keep trying random values (called nonce) until you find a solution
- Like trying random combinations to a lock…
- *The more computing power* you have, **the** *faster* you can solve the cryptopuzzle.
- "*Magic blocks*" are blocks with cryptopuzzles, where everyone has the same power.

Proof-of-Work Example

E.g., the challenge is:

- sha256sum("data:nonce") starts with an "0"
- Normally more complicated than that! (e.g., 18 zeroes)

- P1 wants to send "1:v" to P2

```
kzhang@grey:~$ echo "1:v:118" | sha256sum
9479038ca7543ece09f48e8c77fcea147d7561cac14058199afea18c2f323b8b
kzhang@grey:~$ echo "1:v:119" | sha256sum
79ae2bbac929112a349c2fe7f50210355f4a24683b2dd1ea8f059c9beeed7fd6
kzhang@grey:~$ echo "1:v:120" | sha256sum
002ce3a3b7092d960abf1795a89f70eb0f9ef960036e7d4620cbd3d26d34ffc8
```

- Send "1:v:120" to p2

Proof-of-Work Example

- P2 verifies "1:v:120" is correct (very quick!)
- P2 wants to send "2:1:v:120" to P3

```
kzhang@grey:~$  echo "2:1:v:120:119" | sha256sum
911ab1edf1f331ff423a45fe4c382db30a3f1cf802bb2211df53c80d5798c7ba
kzhang@grey:~$  echo "2:1:v:120:120" | sha256sum
5344a3561673b1481b9cf69493368ca408b1edef67e3f96819c5d1b36cea53ce
kzhang@grey:~$  echo "2:1:v:120:121" | sha256sum
0a908c651e9ec5374976dc8f49a3342a4a789660011551da8871a6cc123c5b57
```

- P2 sends "2:1:v:120:121"
- P3 verifies "1:v:120" *AND* "2:1:v:120:121" are correct
- If P2 wants to send "2:1:w" and fool P3, it needs to find n1 for "1:w:n1" & n2 for "2:1:w:n1:n2"
- If P3 has a way to *detect* that P2 is *doing too much work*, it can detect fraud.

Bitcoin

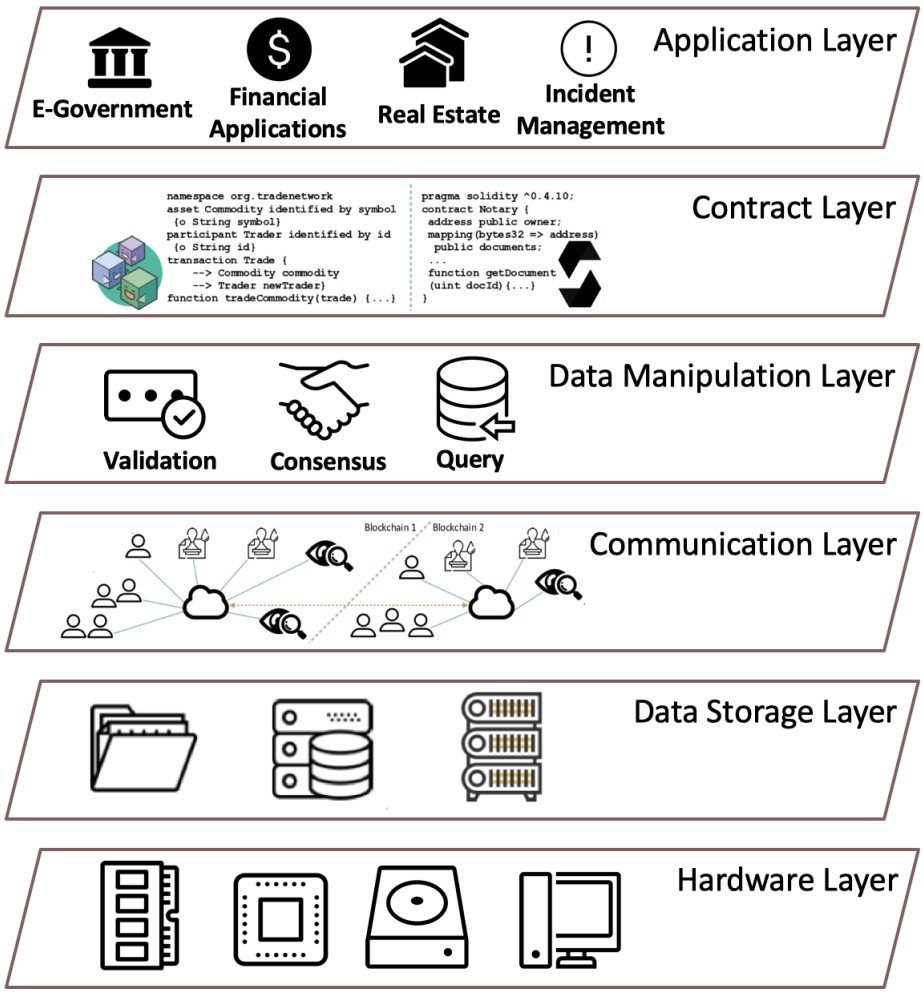
LAYER BY LAYER

Blockchain Reference Architecture

This vision diagram encompasses all aspects related to blockchain technologies.

Upper layers capture application semantics and their implementation.

Lower layers are concerned with technical system details.



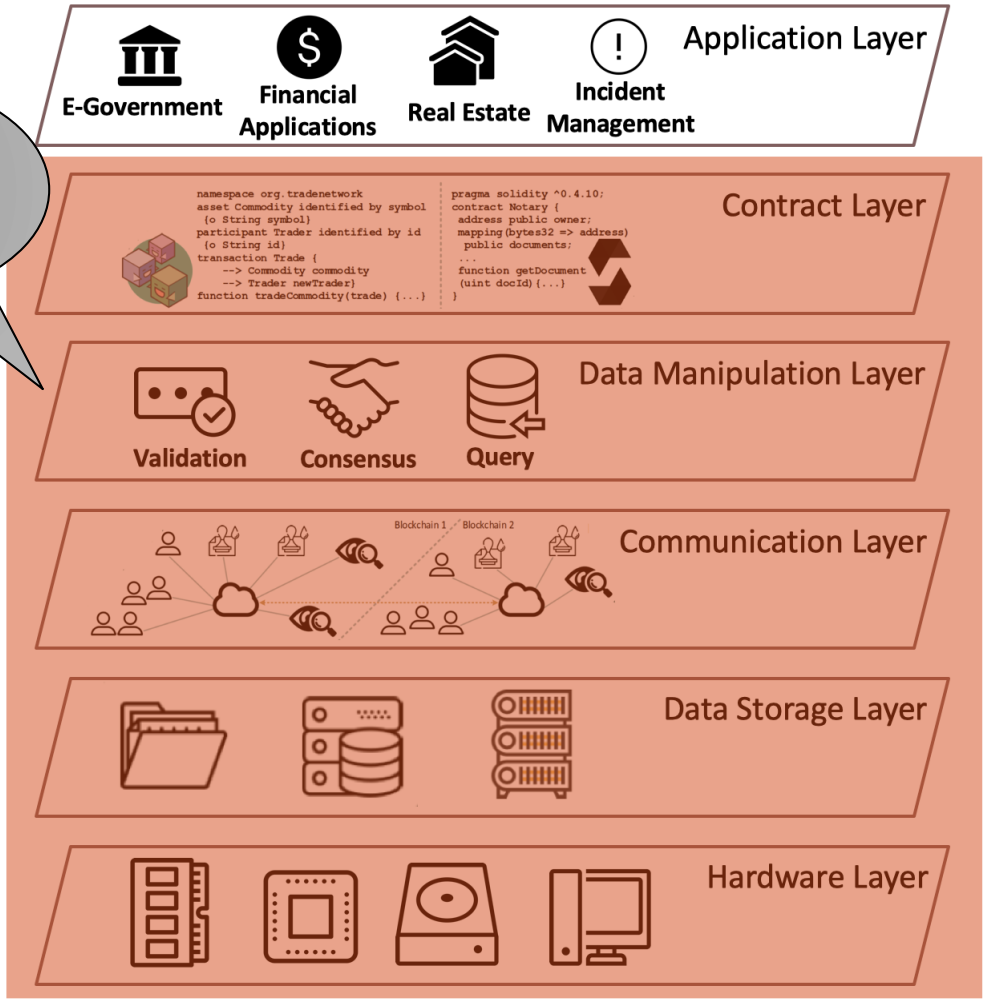
Blockchain Reference Archi e

Bitcoin layers

This vision diagram encompasses all aspects related to blockchain technologies.

Upper layers capture application semantics and their implementation.

Lower layers are concerned with technical system details.



Bitcoin Transactions

Each user possesses a wallet identified by *public/private key* pairs

Transaction A
in
out 1
out 2
฿1 -> Alice

Transaction B
in 1
out 1
฿2 -> Alice

Transaction C (by Alice)
in 1
out 1
฿2 -> Bob
out 2
฿0.9 -> Carol
in 2
out 3
฿0.1 -> _

User encrypts a new transaction C using the private key

C contains outputs to wallet addresses

Tx C must reference *unspent transactions outputs* (UTXOs) from previous blocks equal to the total output of tx C (3 BTC)

The *transaction fee* is given as reward (*explained later*)

Once spent, a TXO cannot be used again: miners *verify* every transaction

Wallets and addresses



Users require a wallet to store money

- This includes any user, **including but not limited to** miners

Wallet is authenticated and identified by a public/private key pair

- Generated using ECDSA (Elliptic curve cryptography)
- More details here: https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses

Redeeming transactions:

- Each TXO address is a hash of the public key of the receiver, who signs proof with the private key
- Transactions do not have a "from" address, so it is impossible to prove you are the sender
- Each address is designed to be single use: wallet programs will automatically generate new addresses

Losing your private key:

- Loss of private key means the wallet and its funds are permanently locked, as it is no longer possible to sign proofs redeeming existing TXOs.
- This money is essentially lost, thereby reducing the total amount of currency in Bitcoin
- Trusting an online service to store your key is also risky, since there is no way to prove that you are the rightful owner if the key is stolen or misused
- The most reliable solution is to store your private keys on tamper-proof hardware wallets

Communication in Bitcoin

Broadcast to all the network

Two primary uses

- Users broadcast their transactions
- Miners broadcasts updates to the blockchain (new blocks)

Implemented via gossiping protocol in a P2P network

- Not terribly efficient but has not been a bottleneck so far

Works because financial transactions are very short and their rate in Bitcoin is far below that of credit cards

Needs to be fairly reliable for the system to work but 100 percent reliability in message delivery is not required

- Users and miners need to detect message loss and retransmit messages if needed

Message propagation should be reasonably fast

- Slower network quantifiably increases the risk of attacks

Transaction Flow



Alice
(Sender)



Bob
(Receiver)

1. Bob generates and send a public key address.
2. Alice creates a transaction using this address.
3. Alice sends the new transaction to the network.
4. The transaction is broadcast using gossiping.
5. The transaction is included in a block.
6. Bob can verify the transaction is in the blockchain.
7. Bob can now sign new transactions which redeem this address.

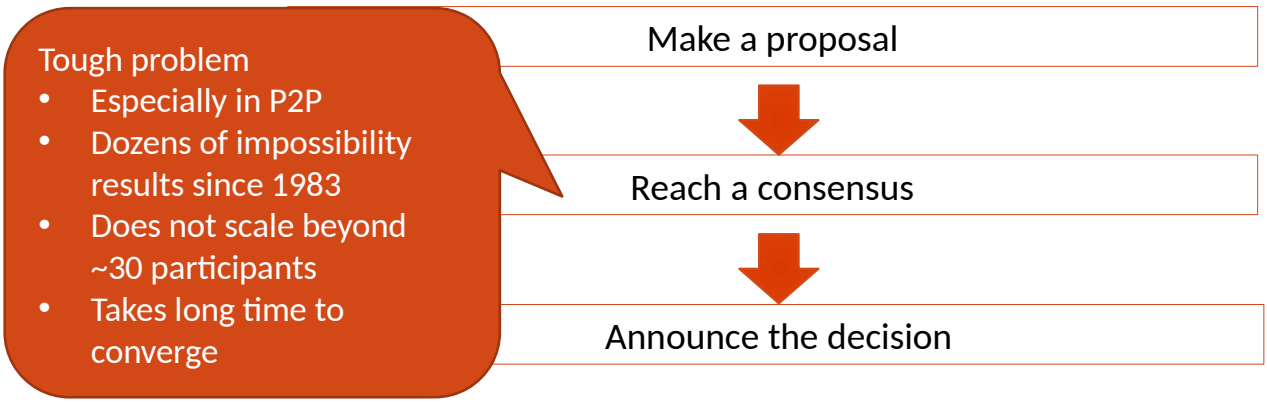


Consensus in Bitcoin

The network needs to agree on

- Which recently broadcast transactions go into the blockchain
- In what order

The general anatomy of consensus:



Challenge 1: who proposes and when?

The network cannot sustain each and every user or peer making a proposal whenever she wishes

Made worse by the proliferation of identities (Sybil attack)

Need to moderate the number of proposers and rate of concurrent proposals

- While keeping them sufficiently high

Several principal solutions

- Proof-of-work: need to do heavy computation and show the proof of it
- Proof-of-stake: need to possess a sufficient amount of coins

Important optimization: propose new transactions in batches

- A block in Bitcoin is structured as a tree of proposed transactions
- With nice cryptographic properties; called a Merkle tree

Cryptopuzzles in Bitcoin



The proposer has to find ***nonce***, such that *hash*(***nonce*** | *H* | Tr_1 | ... | Tr_n) < ***target***

Effectively has to scan the entire ***nonce*** space

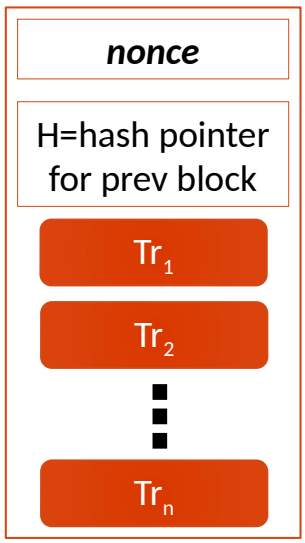
target is a fraction of the hash space

- Every node recomputes ***target*** every 2016 blocks
- Such that the average time for the whole network to solve a cryptopuzzle is 10 min

For proposer *p*,

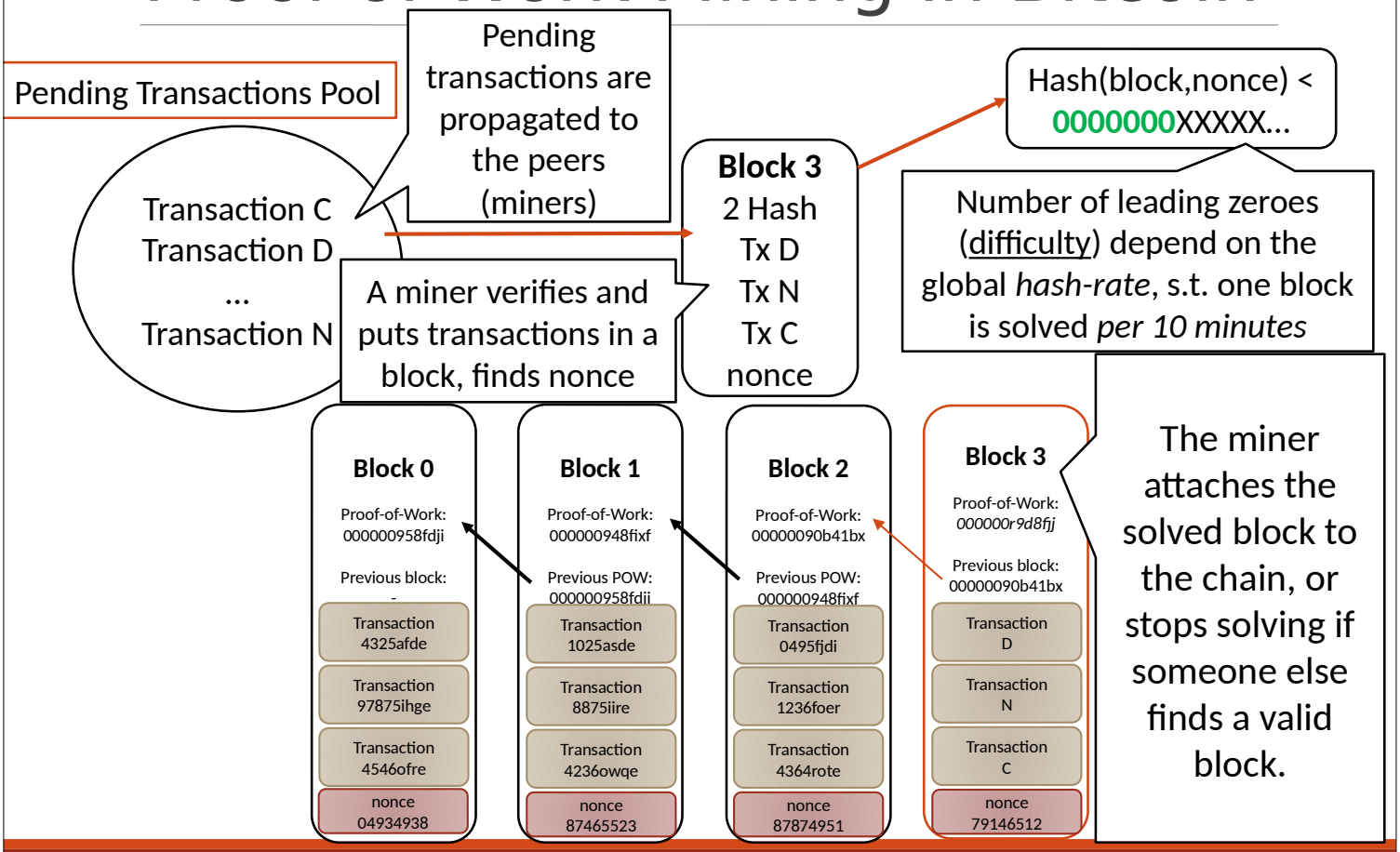
$$mean\,time\,to\,next\,block = \frac{10\,minutes}{fraction\,of\,p's\,computing\,power}$$

The solution is fast to verify



A block in Bitcoin

Proof-of-Work Mining in Bitcoin



Challenge 2: Why propose non-empty blocks?

Two incentive mechanisms in Bitcoin

- Block creation reward: a block proposal creates a number of new bitcoins and transfers them to the proposer
  - Included as a separate transaction in the block
  - Ensures that each proposer solves a different cryptopuzzle
  - The only way to create new bitcoins
  - The amount is predefined and gets halved every 210,000 blocks
  - Predicted to go down to zero before year 2140
  - The geometric progression totals to 21 million bitcoins
  - The rules may change in the future
- Transaction inclusion fee: Alice can decide to pay a small fee to the block creator as part of her transaction
  - Voluntarily, there is no predefined amount

Cryptoeconomy of Mining

Incentives give rise to the mining industry in Bitcoin

- Miners: cracking cryptopuzzles and listening to transaction broadcasts

Expenses: *mining rig + operating costs (electricity, cooling, repairs)*

- Paid in real currency
- Operating costs are variable

Profits: *block reward + transaction fee * # of transactions in a block*

- Paid in Bitcoins
- The fee and rate of transactions are unpredictable
- The mean time to next block is easy to compute
  - However, the per-miner sample is small while variations are huge

Mining pools: groups of cooperating miners

Reaching consensus in Bitcoin

A miner broadcasts the proposed block

- The block includes a hash to the latest block known to the miner

When a peer receives a proposed block

- Check that the proof of cryptopuzzle solution is valid
- Check that each transaction is valid (business logic)
- If the hash pointer is valid, append the new block to the local copy of the blockchain
- Conflict resolution: if the proposed chain is longer than the current local copy, replace the local copy

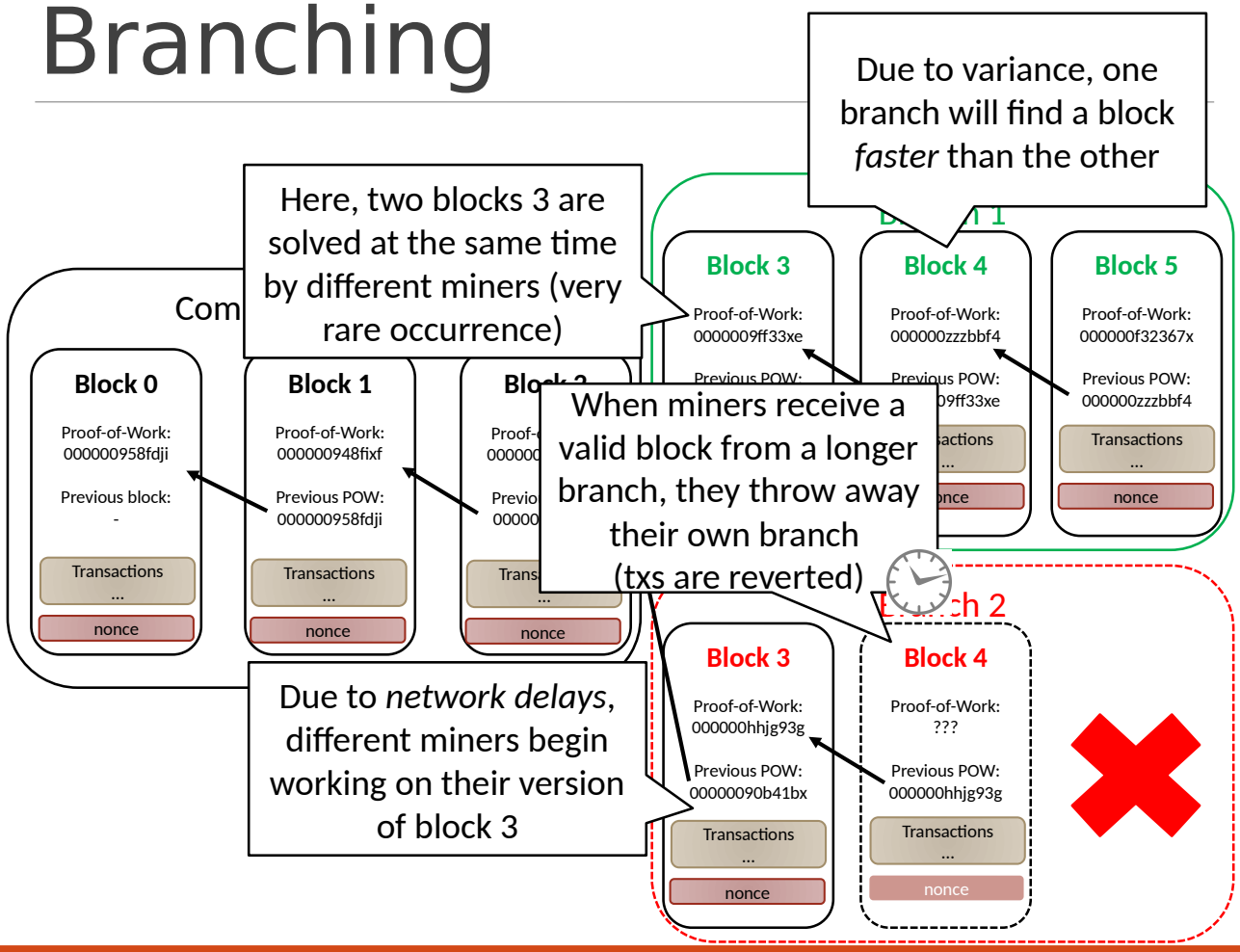
Local copies may diverge!

- Lost messages and concurrent blocks arriving in reverse order
- The probability depends on the network

Probabilistic convergence over time is proven when using the longest chain for conflict resolution

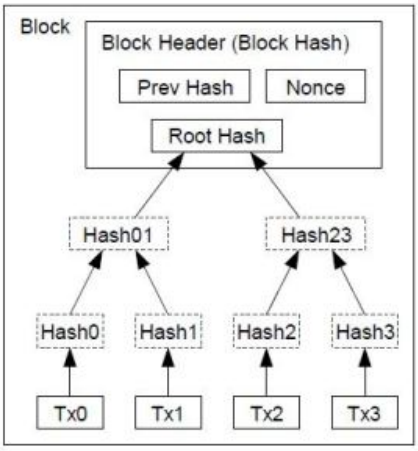
- The probability of a block being non-final decreases exponentially with the number of later blocks stored in the chain
- The standard client sends a confirmation after six later blocks stored in the chain
- Takes an order of one hour in practice

Branching



Data Structure within a Block

Merkle Tree



- To avoid hashing the entire block data when computing PoW, only the *root hash* of the Merkle tree is included.
- For users without a full copy of the blockchain, *simple payment verification (SPV)* is used to verify if a specific transaction exists.
  - A *Merkle proof* only requires the transaction itself, block root hash, and all of the hashes going up along the path from the transaction to the root, e.g., Hash01, Hash2 (for Tx3).
- Spent transactions can be *pruned* in the local copy, leaving only the necessary intermediate nodes to save space.
  - E.g., if both Tx0 and Tx1 are spent, we can prune everything under Hash01

Data manipulation and queries

Reading the ledger and verifying its correctness is straightforward but time-consuming

- Publicly available, no access control whatsoever
- A copy is held by many users (over one million today)
- Users are encouraged to download and run a verification

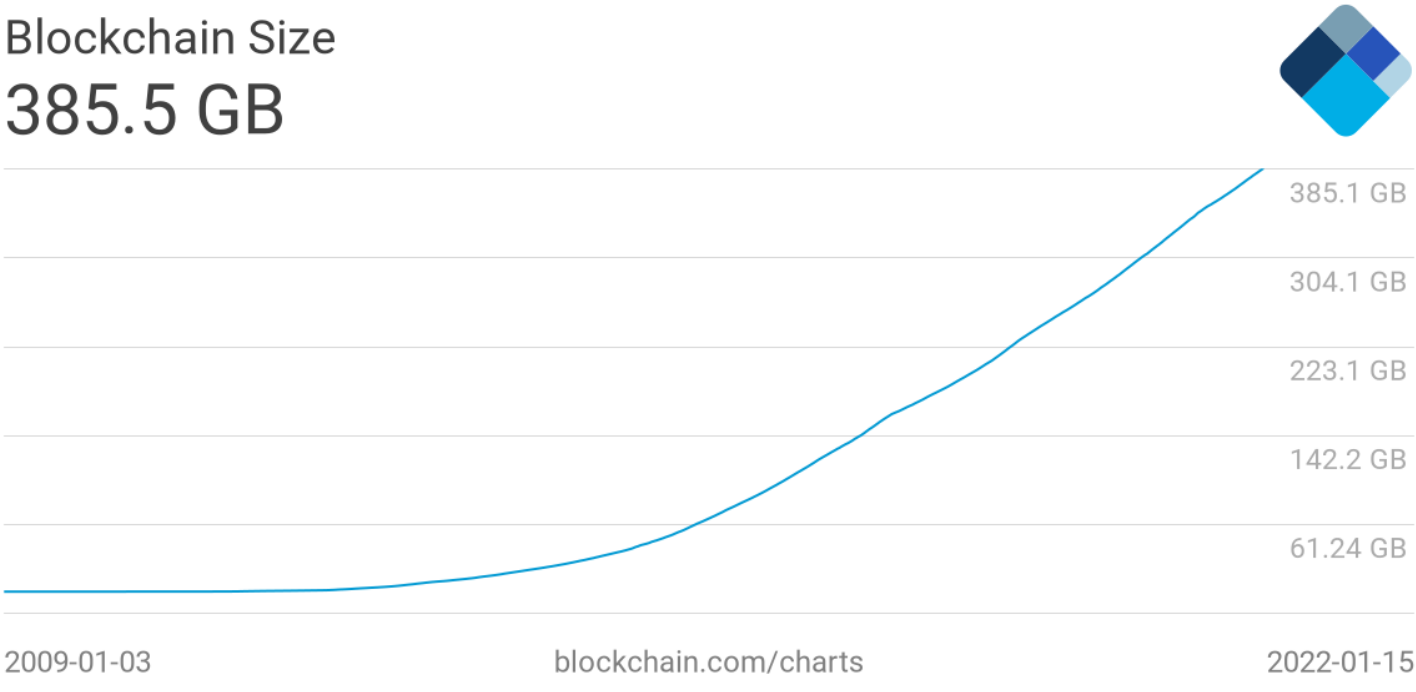
Transparency is a boon for data integrity but a bane for privacy

- Public keys are used as user identities
- A key can serve as a pseudonym, difficult to link to real identity
- A user can use a different pseudonym for each transaction
- The main threat comes from analyzing the history of transactions and linking them together

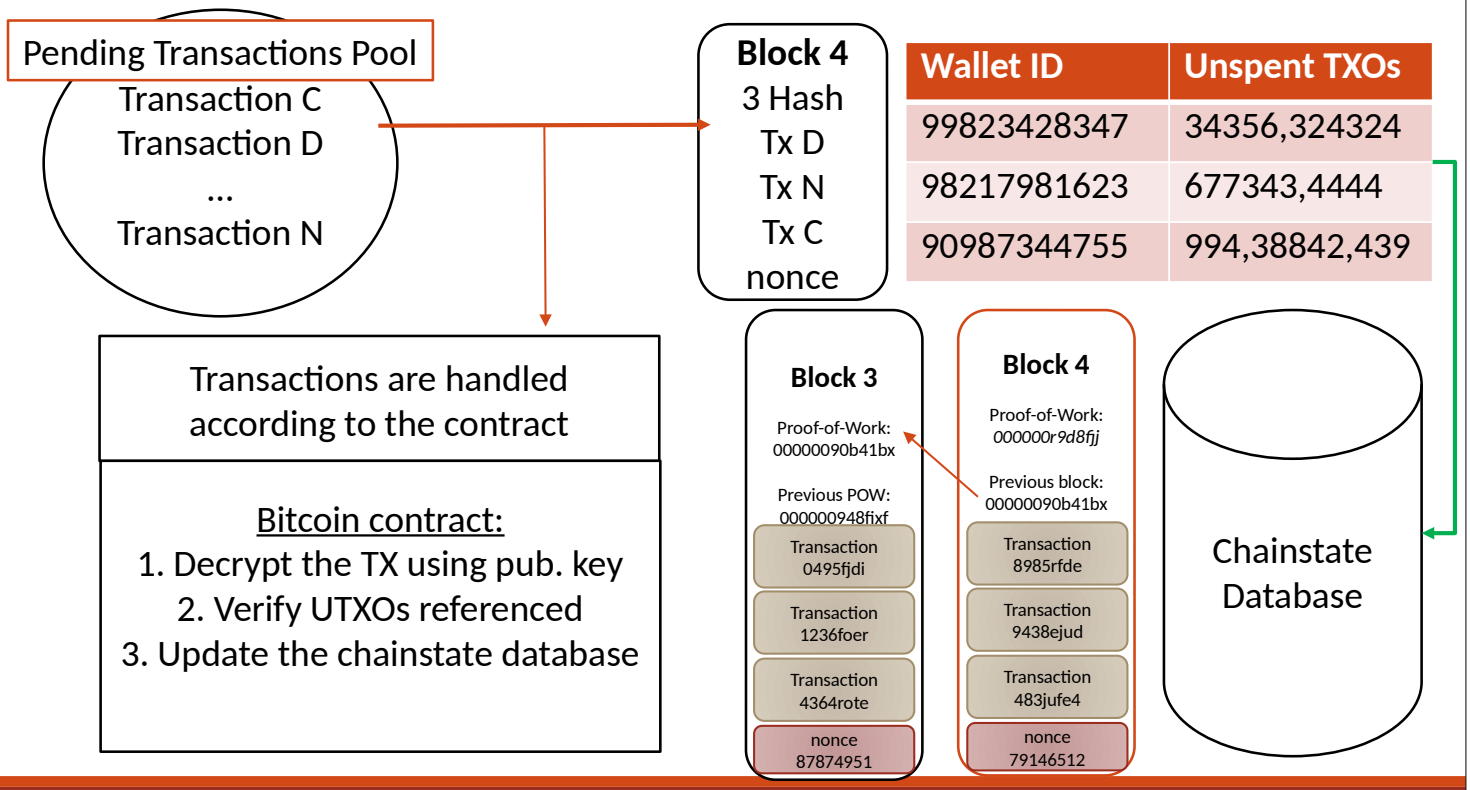
Temper-resistance is mostly a blessing

- But also a curse: difficult to compact or prune the history

Size of ledger



Bitcoin's "contracts"



Business logic in Bitcoin

The output additionally includes a verification script

- representing the conditions under which the output can be redeemed, i.e., included as an input in a later transaction
- A typical script: "can be redeemed by a public key that hashes to X, along with a signature from the key owner"

There is also a redeeming script attached to the input

Both scripts are executed by whoever verifies the redeeming transaction, such as a proposer

A script language with an order of 200 commands

- Support for cryptographic primitives
- Rather ad-hoc

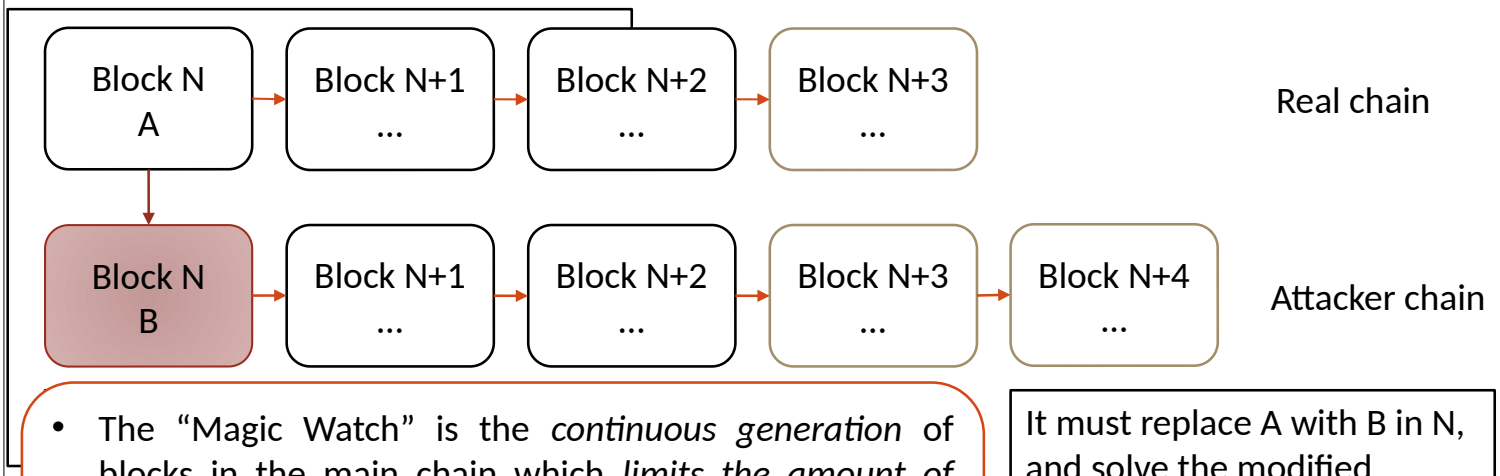
Preventing double spending

(51% Attack)

Transaction A
฿1 ->
Merchant 1

Transaction B
฿1 ->
Merchant 2

A malicious attacker creates two transactions using the same money (*double-spending*)



- The "Magic Watch" is the *continuous generation* of blocks in the main chain which *limits the amount of time* an attacker has to create its own chain.
- If the attacker owns *>51% of the power* in the network, the "Magic Watch" gives *enough time* to the attacker to *tamper the data*!

It must replace A with B in N, and solve the modified puzzles for the blocks faster than the real chain grows so that it can become longer



Other attacks (cursory)

Stealing bitcoins is hard because of digital signatures

- If, however, someone accumulates a lot of bitcoins, it becomes a prime target

Denial-of-service on the entire Bitcoin network is hard because of proof-of-work

- Still possible to bombard the network with invalid transactions

Starving a specific user: does not work if there is a sufficient number of honest miners

- Possibility to blackmail users with high tx fees if miners are "rational"
- cf. feather forking attacks

Economic attacks: selfish mining

- Attempts to maintain private branches longer than the public branch
- Releasing a longer private branch causes honest miners to lose revenue, "stolen" by the attacker
- 25% attack with "rational" miners

Limitations of Bitcoin

Limited expressiveness

- Cryptocurrency only
- Each app requires new platform (e.g. NameCoin, PrimeCoin, CureCoin)

Slow block time (10 mins)

- Also slow confirmation time (1+ hour for 6 confirmations)

Hard/Soft forks

- Updates to the code cause forks
- Hard forks are not compatible
- Duplicated money
- Bitcoin: Cash, Classic, Gold

Slow transaction rate

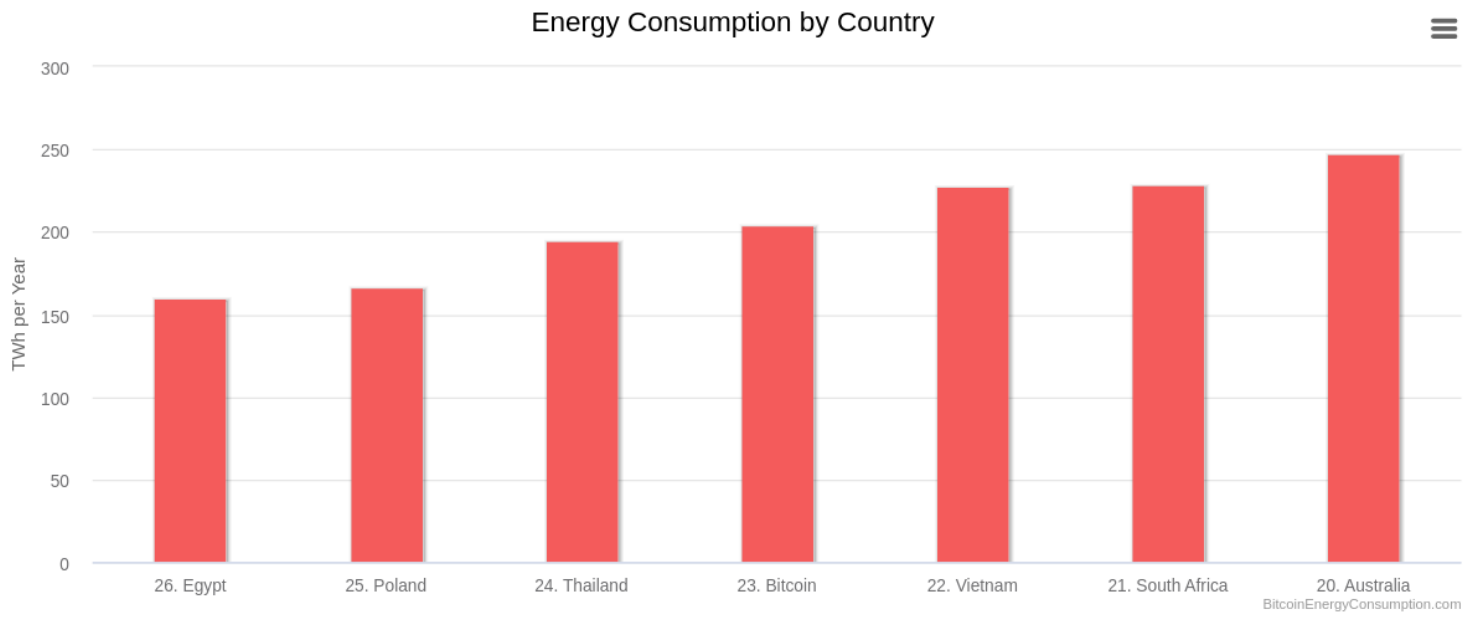
- 7 transactions/second
- VISA Network: 2000 tps (average)
- Limited block size (Segwit2x: 1MB -> 2MB)

Weaknesses of proof-of-work

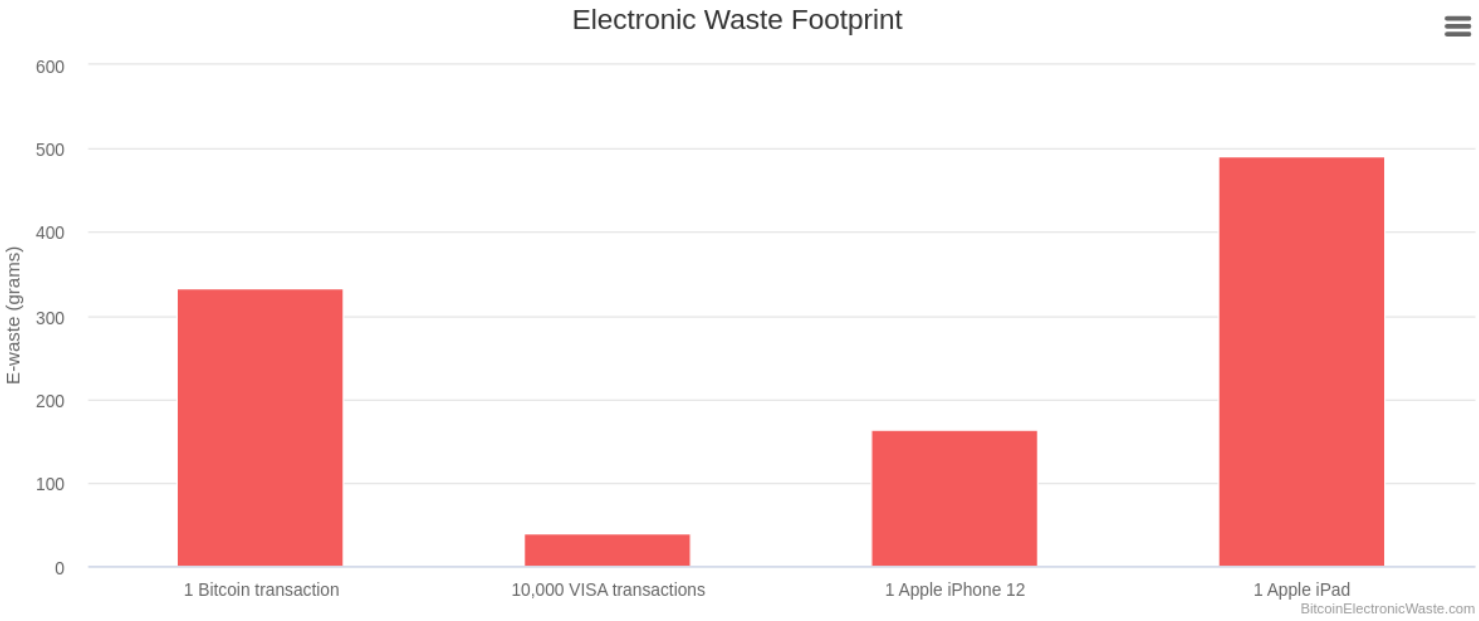
- Environmental impact: ~1000x more energy than credit card
- Comparable in energy consumption to a decent size country

Long bootstrap time for a miner

- Full ledger: 164 GB (2018/04)
- CPU/IO cost to verify each transaction/block
- Takes hours/days



Bitcoin vs. VISA



Blockchain Systems

ETHEREUM

HYPERLEDGER



Managing entity: Ethereum Foundation

- Major players: Deloitte, Toyota, Microsoft, …

Focus: Open-source, flexible, platform

- Cryptocurrency: 1 Ether = 1e18 Wei (502 USD, 2018/04)
- Smart contracts: Solidity, Remix (Web IDE), Truffle (Dev./Test), *Viper*
- Ethereum Virtual Machine (EVM)
- Permisionless (public) ledger: Proof-of-Work, *Proof-of-Stake (Casper)*
- Limited support for permissioned ledgers

Notes

- GHOST Protocol: Merging of branches
- DOA Event: $150 million lost, hard forked into Eth. Classic
- *Scalability: Sharding and Plasma*

Evolution in business logic

Proliferation of Bitcoin spawn-offs

- Digital currency is not the only electronic object of value
  - Documents: authorizations, legal, diploma, design, various deliverables
  - Software
- Support for extended financial applications such as crowdfunding
- Support for multi-party escrow transactions

Ethereum envisioned that a single platform supporting the above is better than hundreds of specialized systems

- Provided a verifiable Turing-complete script language
- With script templates
- Scripts can be stateful, with a state stored on the chain

Benefits of smart contracts

Compared to a human intermediary

- Cheaper
- Open and transparent program that fulfils the contract and does nothing else
  - Does not peek into your data
- Highly resistance to attacks

Compared to distributed databases

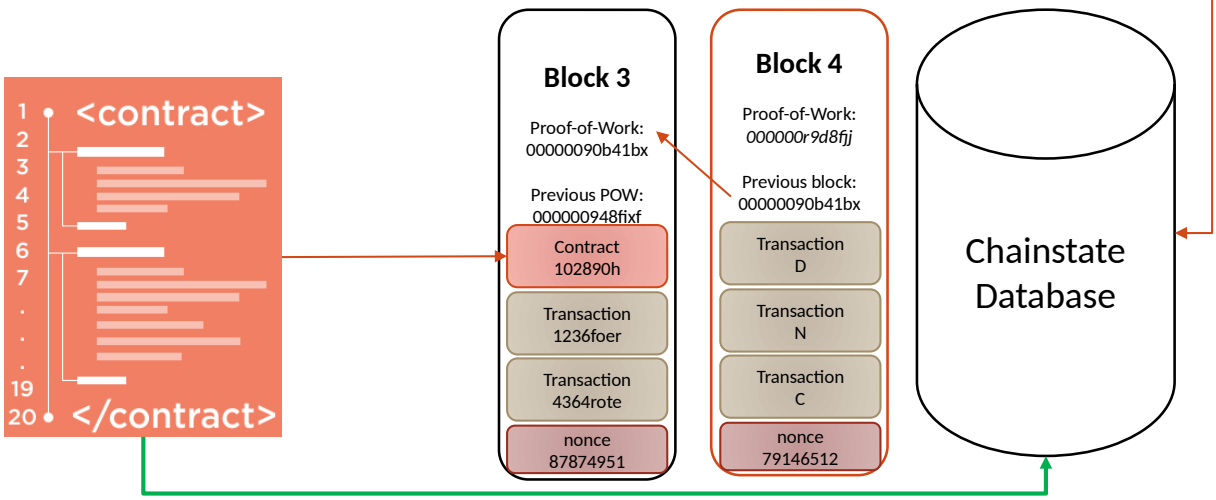
- Rule-based rather than data-based
- Rich language and (relative) easy of development
- The collection of rules is transparent and reusable
- May initiate and play an active role in the communication
- May integrate and fuse data from multiple sources

Smart Contracts

- Contracts contain *executable bytecode*
- Created with a blockchain tx
- Contracts have internal storage

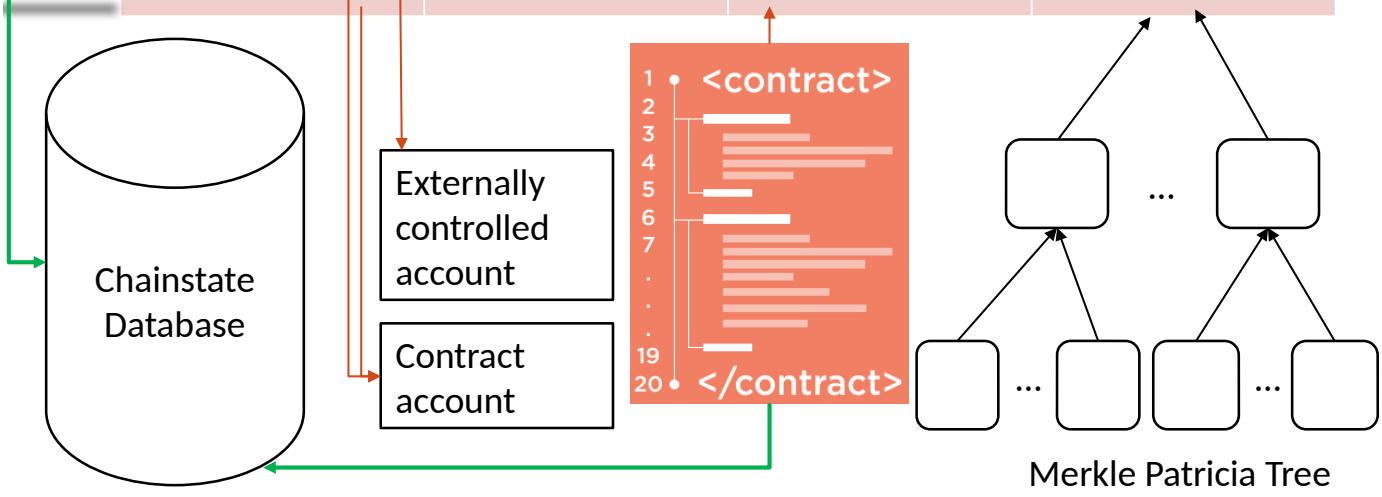
Contracts execute when triggered by a transaction (or by another contract)
Execution time is limited by *gas*
Example: Land registry

Wallet ID	Held Titles
99823428347	34356,324324
98217981623	677343,4444
90987344755	994,38842,439

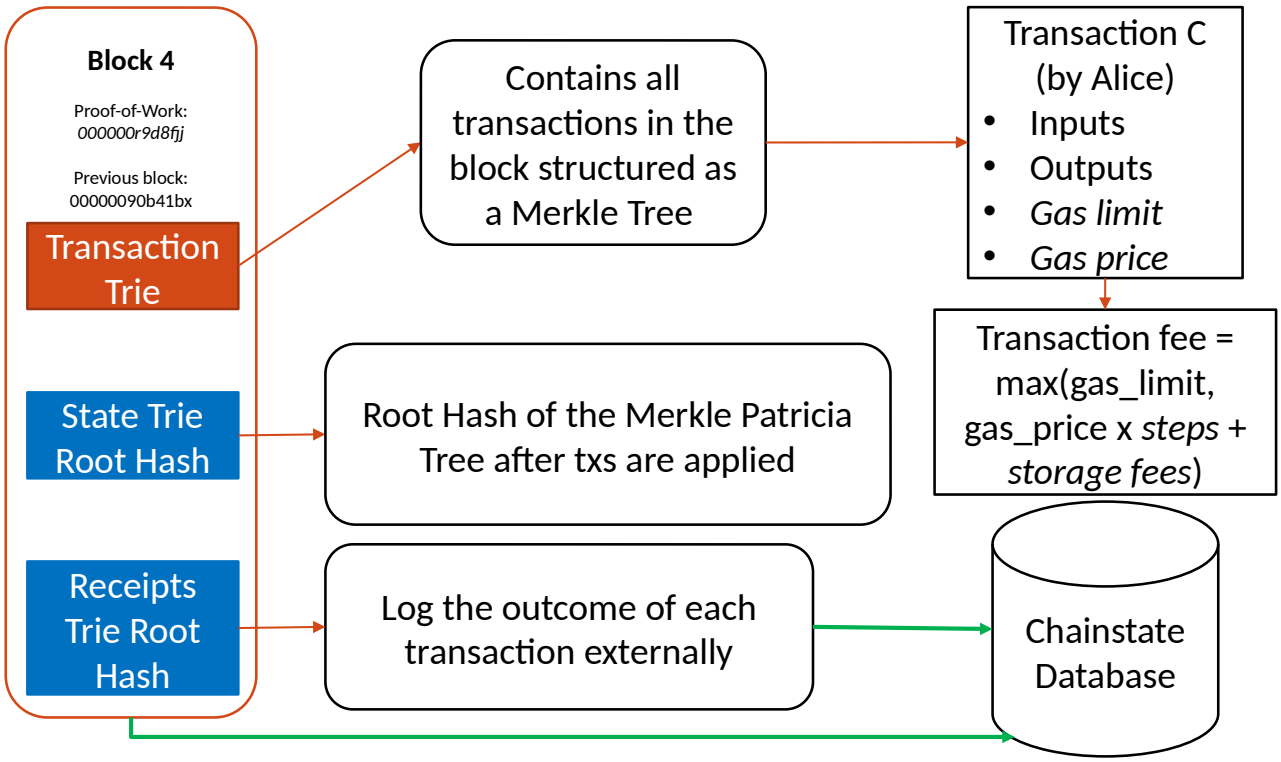


Account State ("World State")

Wallet ID	Balance	Code Hash	Internal State
99823428347	45.12	-	99554HGJ
98217981623	1123.332	9ERU12T4	3453ADFG
90987344755	9.3444	0490CNDJ	132GJR4



Execution and Mining



Comparison with Bitcoin

	Bitcoin	Ethereum
Transactions	Transfer of bitcoins	*Contract creation*, transfer of ether, *contract calls*, *internal transactions*
Accounts	User wallets	Externally owned accounts, *contract accounts*
Transaction fees	Amount specified by sender	Gas calculated using sender's values
Block content	Transactions trie	Transactions, *State Root Hash*, *Receipts Root Hash*
Chainstate Database	World state: UTXOs for wallets	World state, *receipts*, *bytecodes for contracts*
Querying	Simple Payment Verification	Merkle proofs for *events*, transactions, *balance*, etc.


```
pragma solidity ^0.4.18;

contract ProofOfExistence {

    event ProofCreated(
        uint256 indexed id,
        bytes32 documentHash
    );

    address public owner;

    mapping (uint256 => by

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    modifier noHashExistsYet(uint256 id) {
        require(hashesById[id] == "");
        _;
    }

    function ProofOfExistence() public
        owner = msg.sender;
    }

    function notarizeHash(uint256 id, bytes32 documentHash) onlyOwner noHashExistsYet(id) pu
        hashesById[id] = documentHash;

        ProofCreated(id, documentHash);
    }

    function doesProofExist(uint256 id, bytes32 documentHash) public view returns (b
        return hashesById[id] == documentHash;
    }
}
```

Solidity version

More details in session 4:
https://github.com/ramyhardan/proof-of-existence

An event can be listened to by any client

Getters are automatically generated for public vars

Prepends the code to any function

Constructor is executed when the contract is created

The core function of this contract, allows its owner to confirm a document id with a certain content

ProofCreated event is emitted to the blockchain

View functions are read-only and do not cost gas

Using Contracts (geth)

Compile bytecode:

- > solc --bin soliditySource.sol

Compile the Application Binary Interface (ABI):

- > solc --abi soliditySource.sol

Deploy the contract using a “new-contract” transaction:

```
var abi = <abi from solc>;
var myContract = eth.contract(abi);
var bytecode = '0x' + <bytecode from solc>;
var txDeploy = {from:eth.coinbase, data: bytecode, gas: 1000000};
var myContractPartialInstance = myContract.new(txDeploy);

// Mine block containing transaction...

var myContractInstance = myContract.at(myContractPartialInstance.address);
```

Use IDEs such as Truffle to automatically compile, deploy, and test





HYPERLEDGER

Managing entity: Hyperledger Consortium

- Major players: IBM, NEC, Intel, R3, …

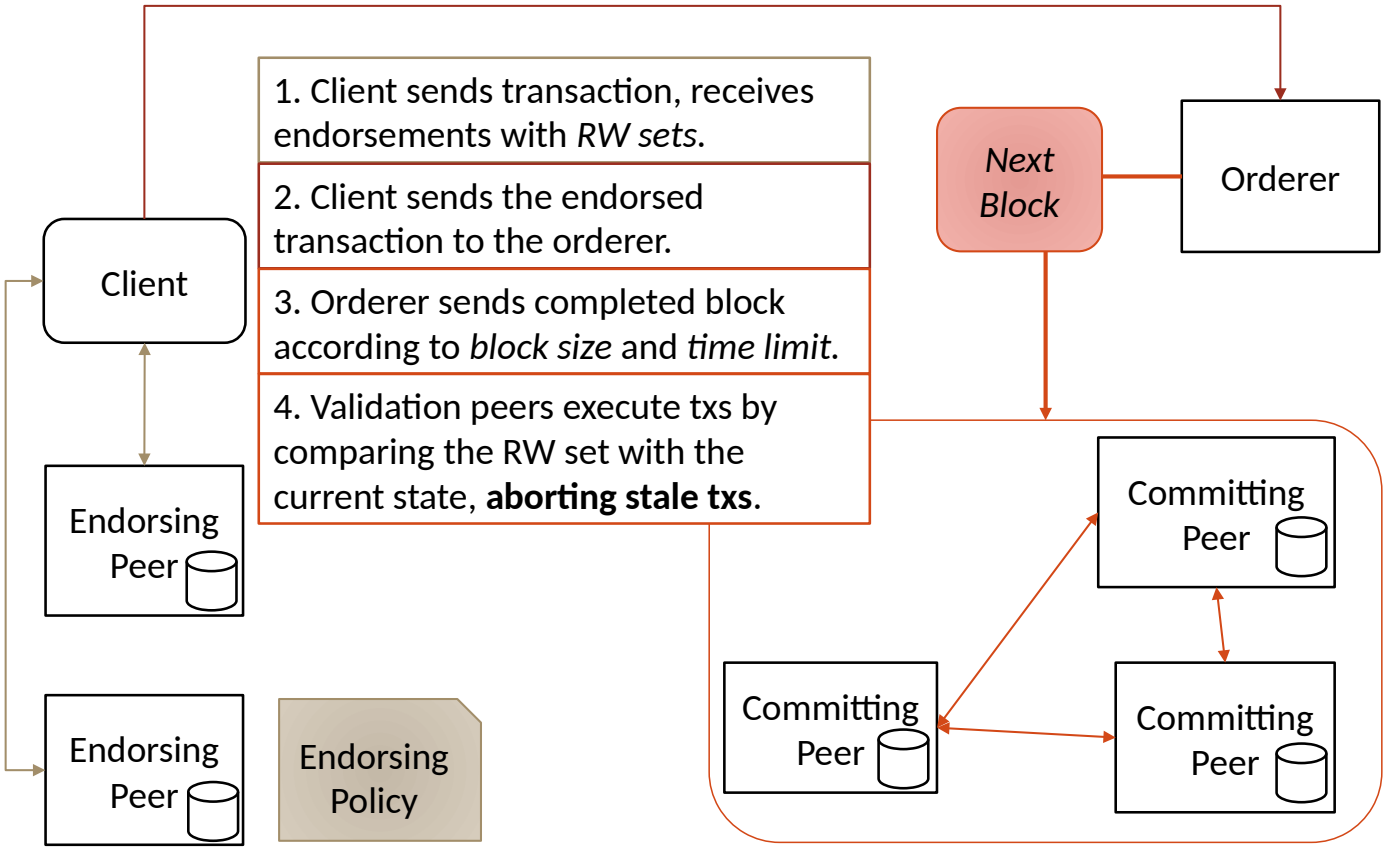
Focus: Enterprise blockchains

- Permissioned ledger (private/consortium network)
- Smart contracts
- Open-source
- World state on CouchDB, event listener

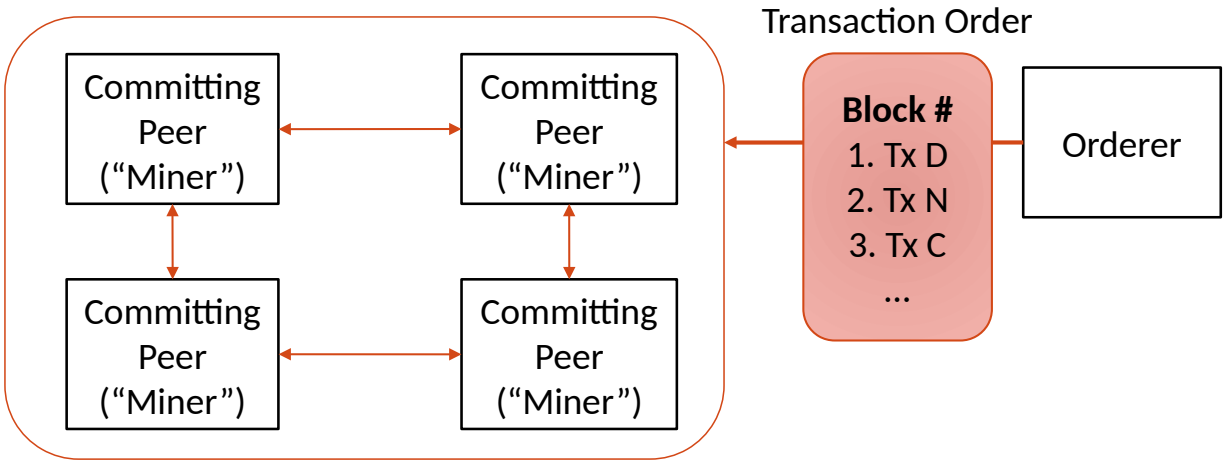
Projects

- Fabric: PBFT Consensus
- Sawtooth: Proof-of-Elapsed-Time (using Intel SGX)
- Composer: Smart contract language and development tool
- Cello: Blockchain-as-a-Service framework
- R3 Corda: Financial applications

Fabric: Transaction processing flow



Fabric: Practical Byzantine Fault Tolerance



HYPERLEDGER

ripple

- Each peer executes transactions in order.
- The resulting block hash is broadcast.
- After 2/3 responses, the block is committed locally (v1.0)

Tendermint

Focus: State Machine Replication (RSM)

- Application BlockChain Interface (ABCI)
- Byzantine Fault-Tolerance Tendermint consensus (up to 1/3 failures)
- Open source

Notes

- Supports 1000s of tx per second with 1000ms latency
- Efficient dynamic Merkle trees (AVL+ Tree)
- Tendermint consensus is used for Burrow, a permissioned version of Eth.

Blockchain Insights

BENEFITS AND CHALLENGES

TAXONOMY OF BLOCKCHAINS

RESEARCH OPPORTUNITIES

Taxonomy

	Anyone can read	Read access restricted
Anyone can propose updates	Bitcoin, Ethereum	Ethereum (Smart Contracts)
Update access restricted	Ripple	Hyperledger, Corda

The above is well defined, but has no common terminology associated with it

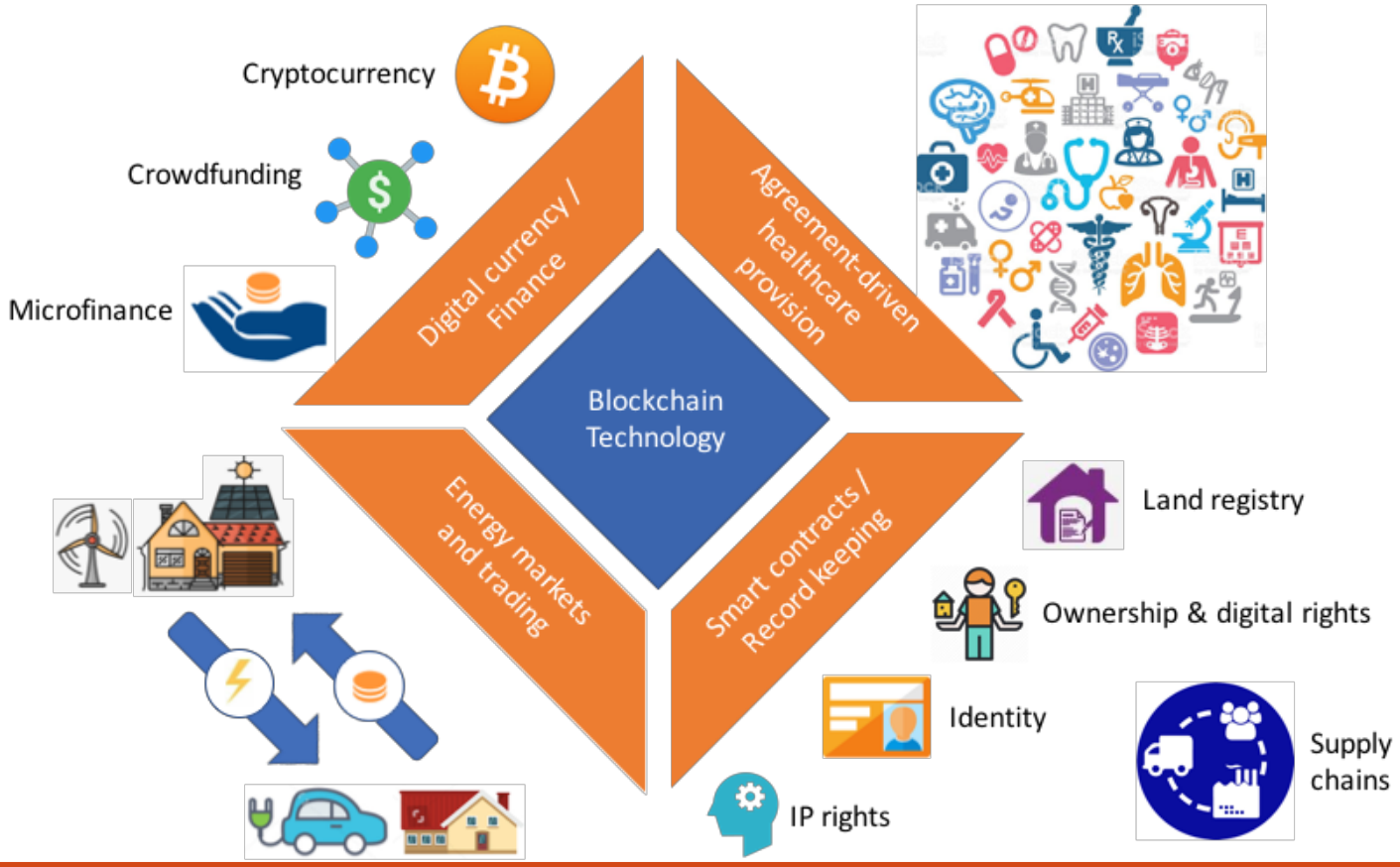
- Journalists use other terms instead: open/closed, permissioned/permissionless, public/private

Decentralization and scale: centralized, large-scale decentralized, and consortium blockchains

Radically different models with respect to

- Identity management and authentication
- Trust and attack model

Versatility and potential



New challenges introduced by DLTs

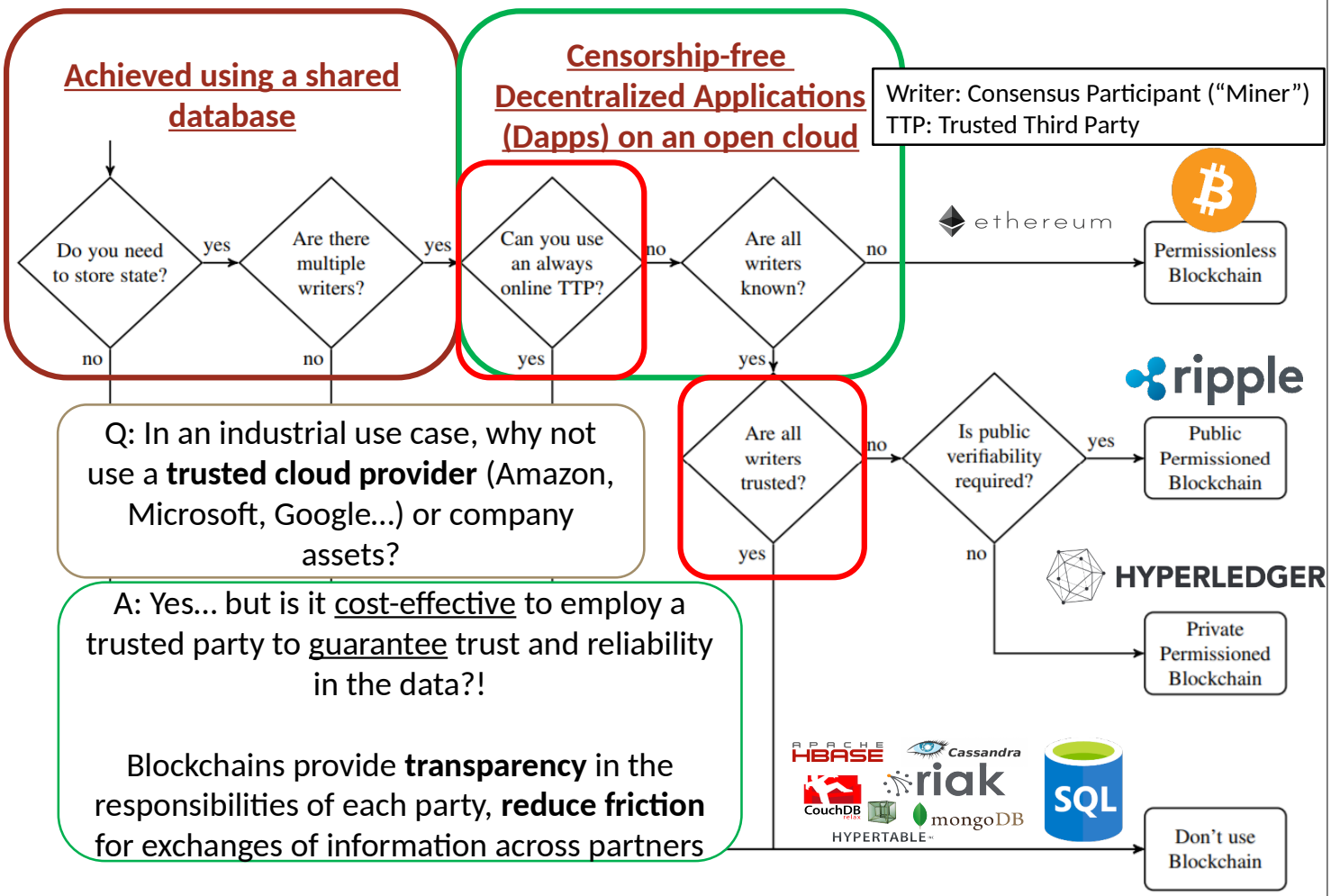
Compared to databases

- Slower
- Lower rate of transactions
- Less compact storage

The technology and even standards (and even terminology) are still developing

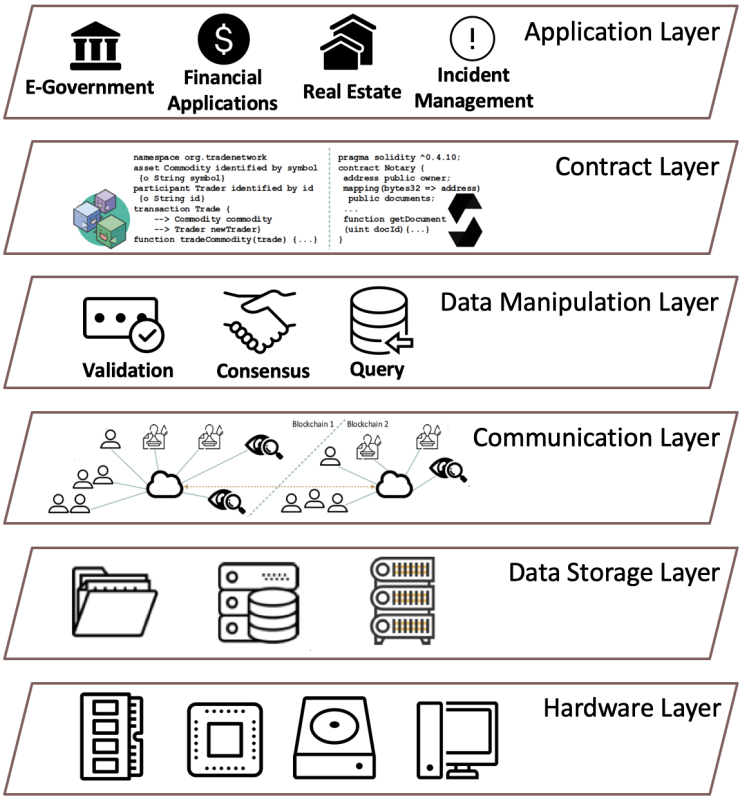
Additional challenges related to smart contracts

- Bug prone, no established programming or verification practices
- State machine execution, with each contract replica performing every action
- If a contracts interacts with an external non-blockchain service, this service needs to be designed with this in mind



Link to our companion papers:
http://msrg.org/papers/bcbi-tr
http://heim.ifi.uio.no/~romanvi/debunking-bc-myths.pdf

Conclusions



- Blockchains provide *decentralized storage and code execution*, and can be used to combat fraud, avoid redundancy, and provide transparency.
- Blockchains rely on *cryptography* and massive replication using a robust consensus mechanism.
- Blockchains are useful for a wide variety of applications, ranging from cryptocurrency to health.
- Research directions exist *across the six layers* for all kinds of applications, and involves different tradeoffs.

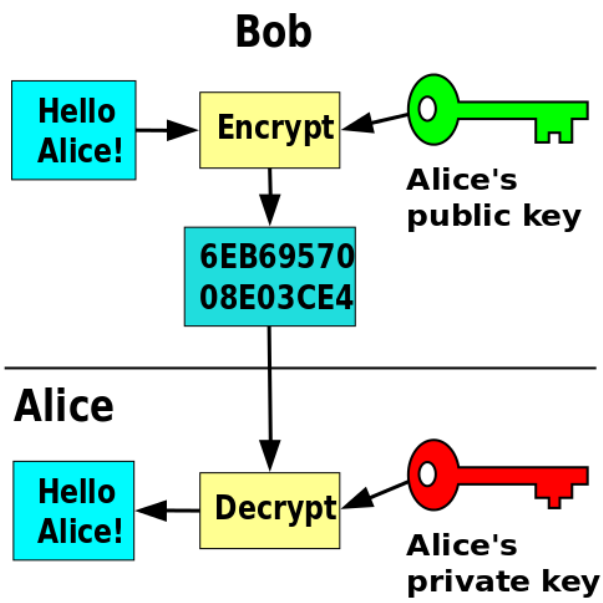
Bonus Material

APPENDIX

Public Key Cryptography

(Asymmetrical Cryptography)

- Recipient's public key is used to encrypt the plaintext to ciphertext
- Recipient's private key to decrypt the ciphertext to original plaintext
- No one can use the public key to decrypt the ciphertext to plaintext



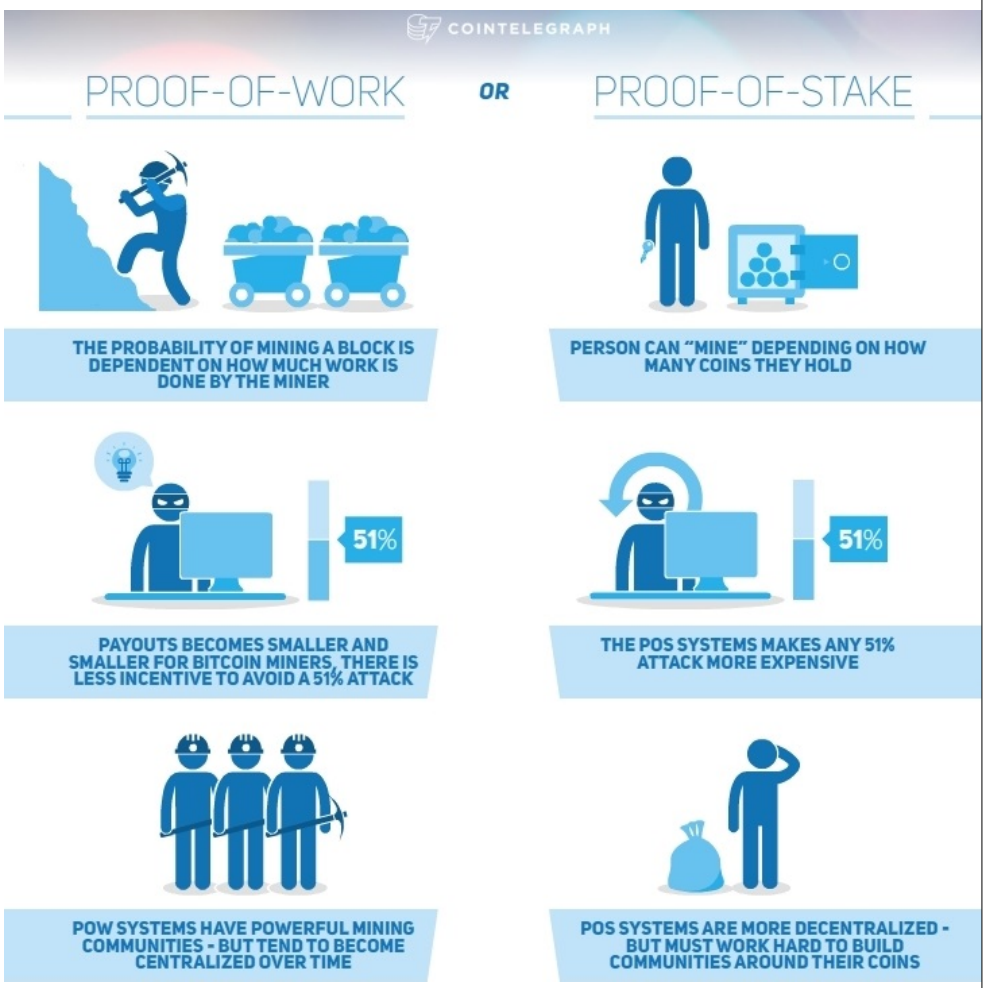
Proof-of-Stake

PeerCoin

Nxt

Ethereum (Future)

"Nothing at stake" problem



Proof-of-Stake Details

verify() function in PoS:

- *sha256(PREVHASH + ADDRESS + TS) <= 2^256 * BALANCE / DIFFICULTY*
- ADDRESS of wallet of the miner, BALANCE is the recorded stake for the wallet
- TS is the timestamp in UNIX time (seconds)
- Thus, only one hash needed per second (per wallet)

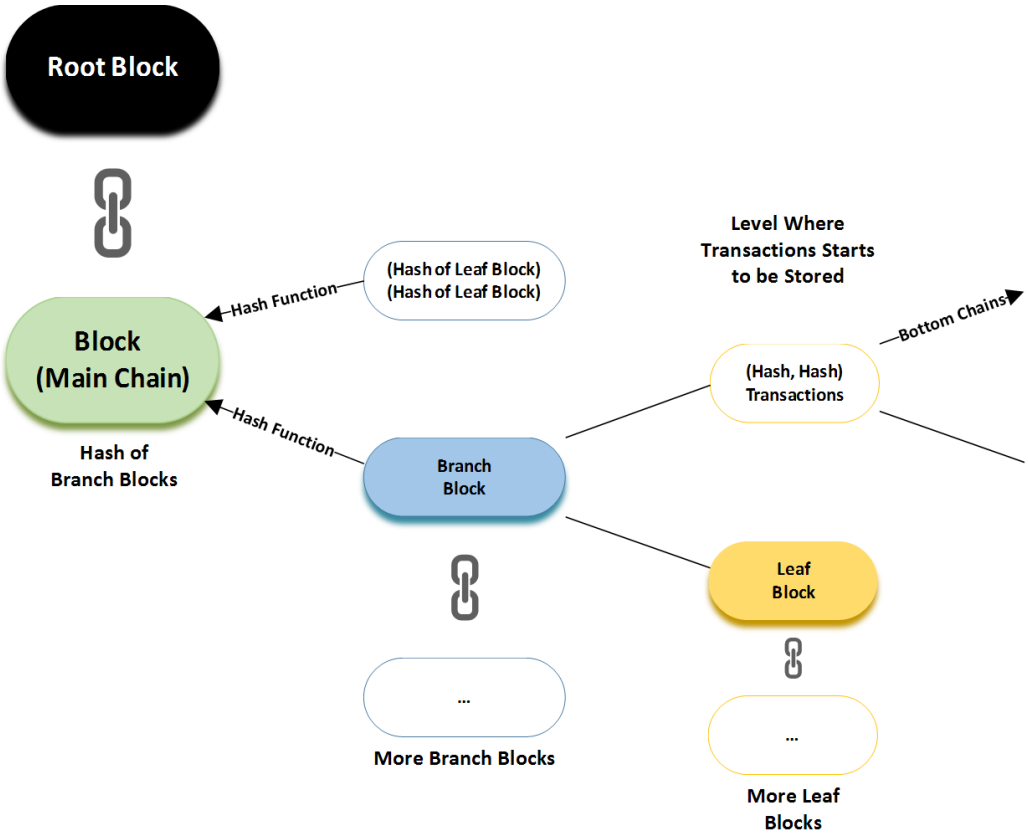
Branches can still exist in PoS:

- Due to propagation delays, multiple timestamps are valid for a block
- The puzzle function does not return an unique winner

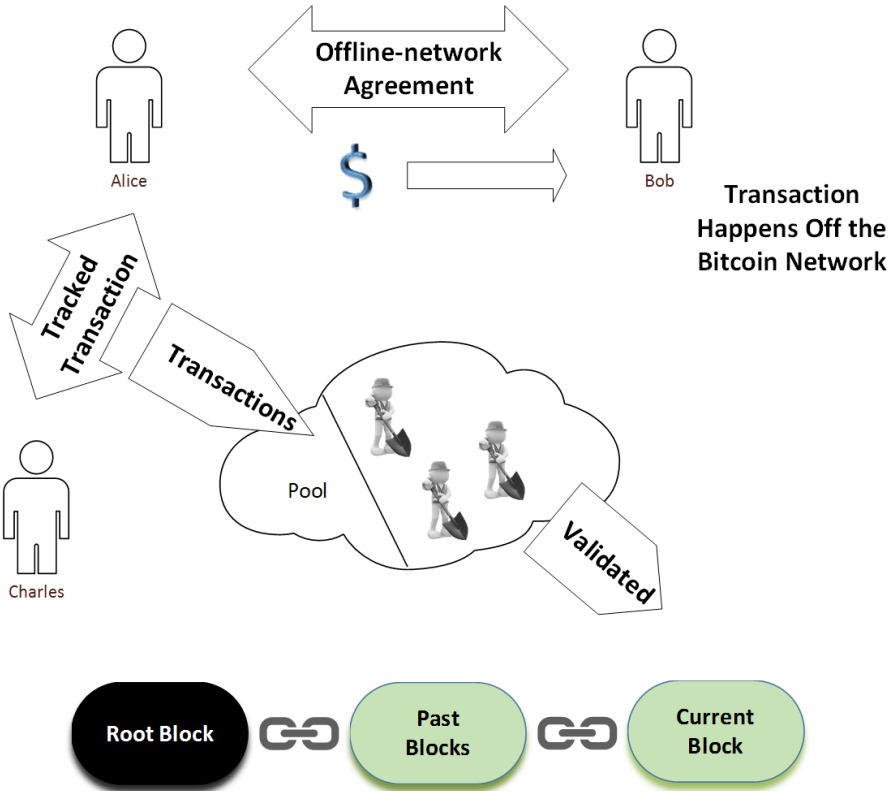
Nothing-at-Stake problem:

- PoW: cannot mine parallel branches since splitting resources is not effective
- PoS: mining parallel branches is easy since it only requires 1 hash/s
- Slasher algorithm: detection of parallel mining confiscates the stake

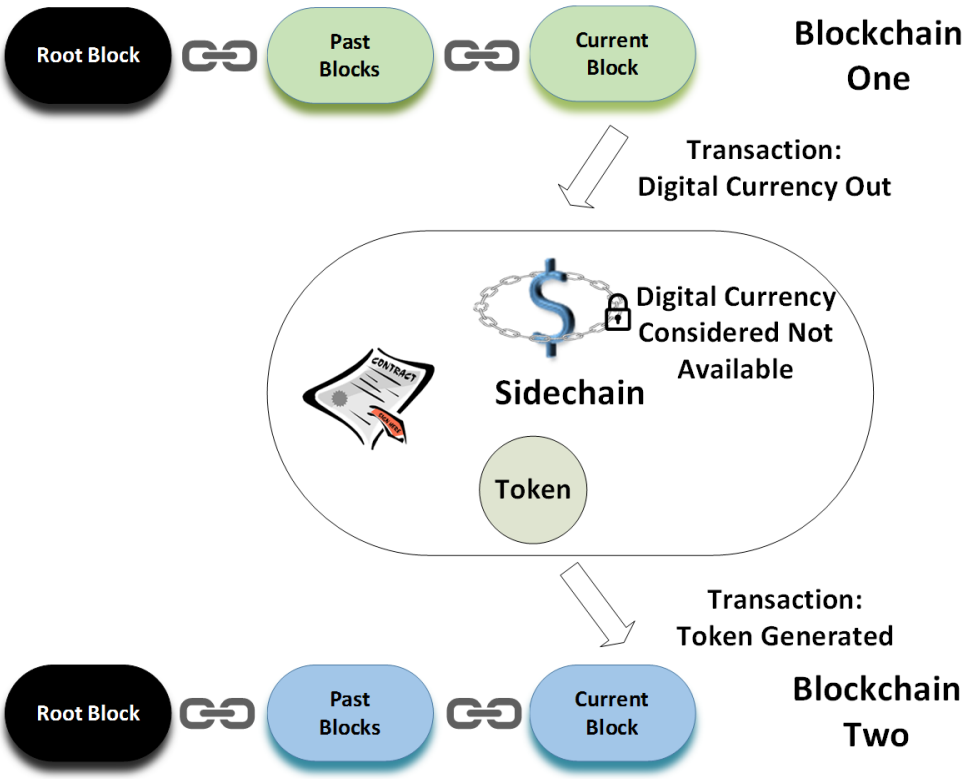
Scalability: Tree Chain - GHOST



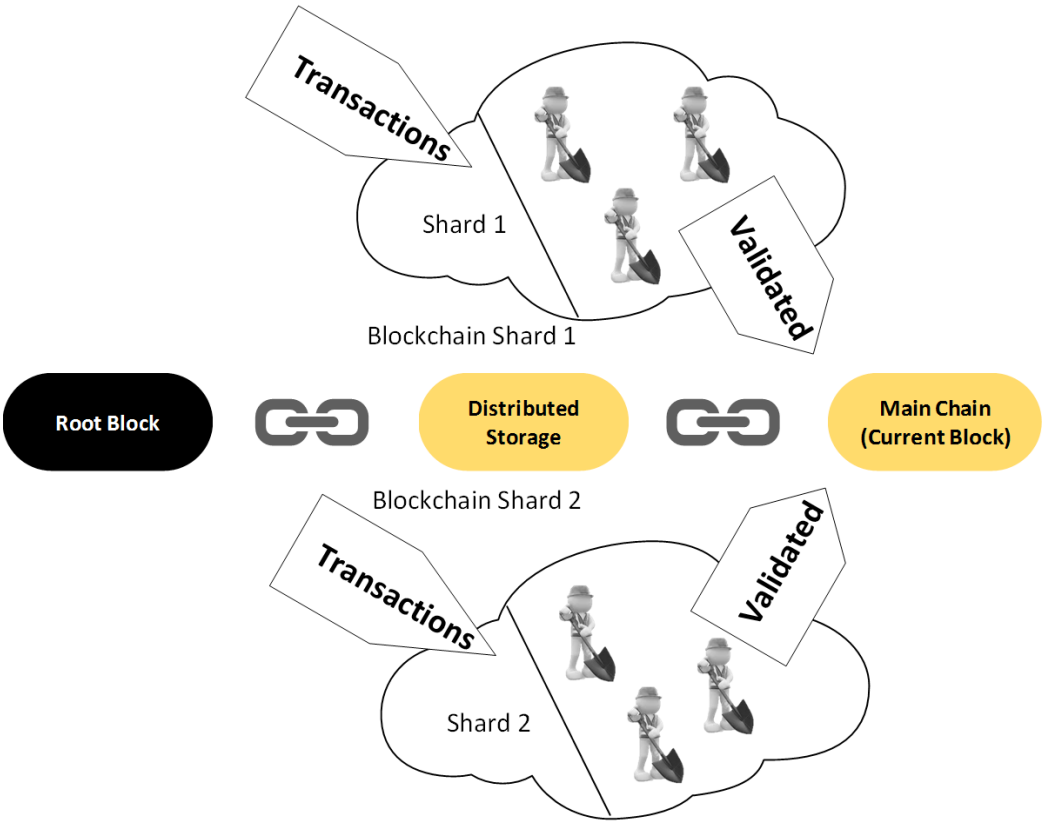
Scalability: Off-Chain



Scalability: Sidechain



Scalability: Sharding



Blockchain Platform

REFERENCE ARCHITECTURE

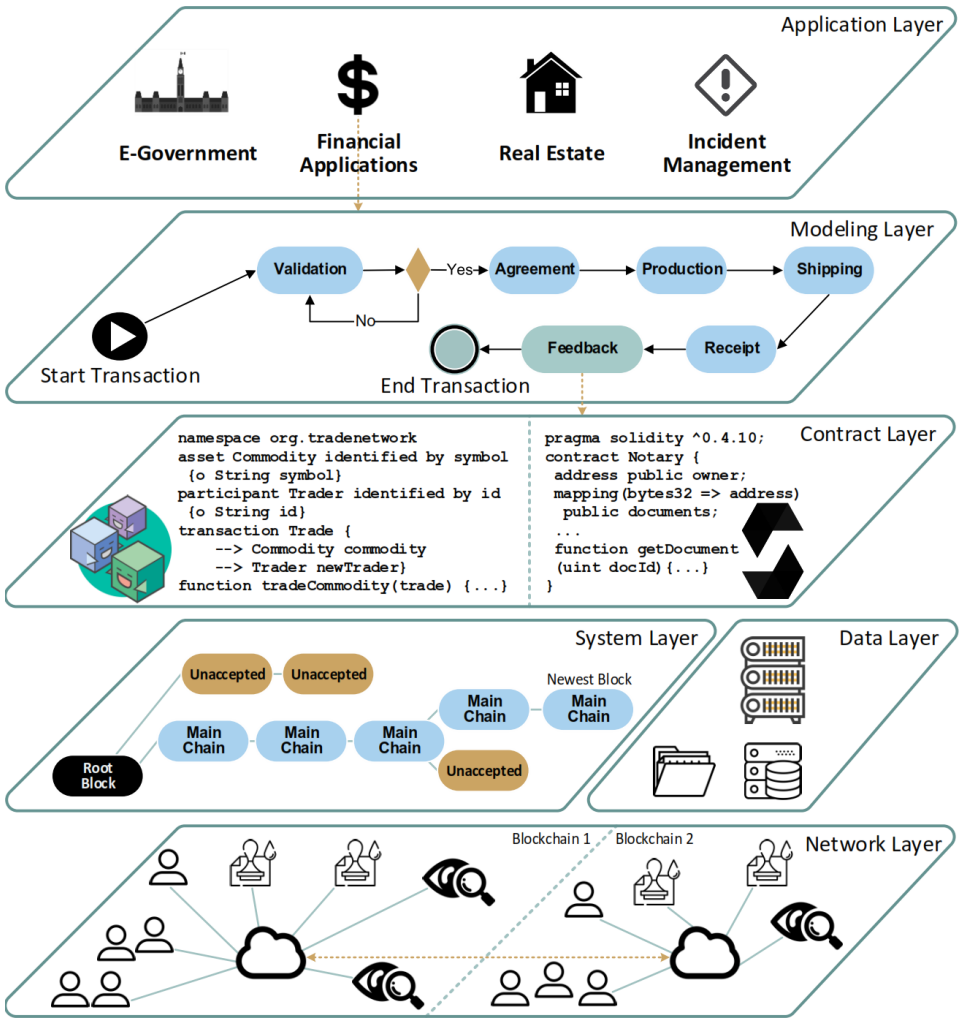
RESEARCH DIRECTIONS

Blockchain Reference Architecture

This vision diagram encompasses all aspects related to blockchain technologies.

Upper layers capture application semantics and their implementation.

Lower layers are concerned with technical system details.



Application Layer

Potential Research Directions

- Identify *application* and *service characteristics* that benefit from a Blockchain-based approach
  - cf. “Do you need a blockchain?” paper
- Develop a *methodology* to evaluate potential applications and select the appropriate *optimized* blockchain system:
  - Position applications as Blockchain 1.0, 2.0, or 3.0
- Create a *standard template* to describe and articulate use cases:
  - Describe actors, assets, transactions, queries, functional requirements, SLAs, etc.

Modelling Layer
Potential Research Directions

- Identify higher-level *modelling* and "programming" abstractions that are useable by business analysts, that are *verifiable*, that offer guarantees to end-users and map these abstractions into lower layers
  - BPMN, Petri-Nets, FSM, Business artifacts with lifecycles
- Identify *common services* and design blockchain *middleware* to support a variety of use cases
  - Identity management (authentication), reputation, risk analysis (spot checks), auditing, bidding, zero-knowledge proofs, document input etc.
- Extend modelling languages using blockchain semantics
  - FSM+: States can be described as *"on-chain"* or *"off-chain"*
  - Use of *Controlled English* which is portable to smart contracts

Smart Contract/Programming Layer
Potential Research Directions

- Design *mappings* for standard modelling languages (e.g., BPMN) into *smart contracts*
  - Create execution engines on blockchains for BPM
- Formally *verify* smart contracts for correctness
  - Use of formal verification tools (e.g., Why3, F*)
- Investigate the use of *domain-specific languages* for smart contracts
  - E.g., to circumvent the halting problem
- *Scalable execution* and storage of smart contracts
  - E.g., Sharding in Plasma, zk-SNARKS in Zcash

System Layer
Potential Research Directions

- Evaluate existing *consensus algorithms* and design new ones specifically tailored to application characteristics with varying tradeoff:
  - *Proof-of-Stake*, *Practical Byzantine Fault-Tolerance* (PBFT), …
- Develop mechanisms to increase the scalability of blockchains:
  - *Off-chain*, *side-chains*, *tree-chains (GHOST)*, *sharding*
- Use of innovative hardware for achieving consensus
  - *Proof-of-Elapsed-Time* using *Intel SGX* (Hyperledger Sawtooth)
- Develop *quantum-resistant* mechanisms for securing Blockchain computations

Data Layer
Potential Research Directions

- Develop effective *data management abstractions* to enable efficient Blockchain computations and verification
  - AVL+ Trees, Merkle Patricia Trees
- Develop *compression techniques* to reduce the size of historical data and scale with the number of users
  - Ethereum Fast Sync
- Provide *off-chain storage (chain state)* which is securely and privately verifiable by the on-chain data, executable by the smart contracts
- Maintain *availability* of smart contracts and assets in the presence of space saving techniques

Network Layer
Potential Research Directions

- Develop effective *networking abstractions* to support *scalable* and *low-latency* blockchain operations
- Investigate effects of *networking characteristics* on Blockchain computations (e.g. network partitions)
- Integrate with *Software-Defined Networking* (SDN) and other technologies
- Tolerate *unreliability* in hardware components (IoT, Edge Computing)
- Support *cross-platform* communication (e.g. private & public networks)