

designed by Startline - Freepik.com

L03-05: Architecture and Technical Basics

Frank Kargl | Blockchain Fundamentals | 2023-10-30



Institut für Verteilte Systeme
Institute of Distributed Systems

What do we need to build Bitcoin (or other Crypto-Currencies)?

Bitcoin aims to build a completely decentralized version of cryptographic cash. This requires some building blocks and poses some challenges:

- **Identities:** owners of values
- **Transactions:** description of value transfers
- **Distributed Database:** persistent record of all transactions
- **Distributed Consensus:** avoiding different views on state of system

Identities in Bitcoin

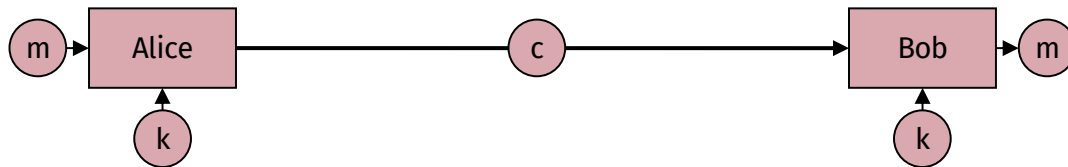
- Identity:
 - We need to be able to distinguish different ‘users’ of Bitcoin and allow them to prove the ownership of value.
 - Users should be able to create and manage identities without help of central authorities.
- Identities are represented by cryptographic keypairs using asymmetric cryptography



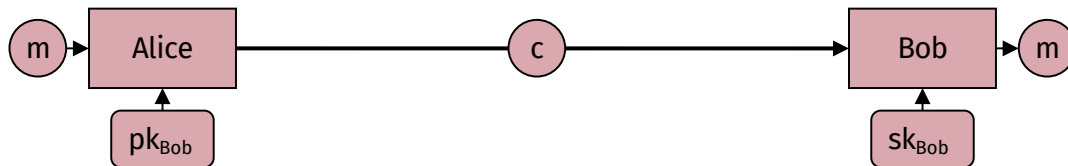
What??!#!?

Terminology for Encryption Algorithms (1)

Symmetric Cryptography (secret key cryptography)

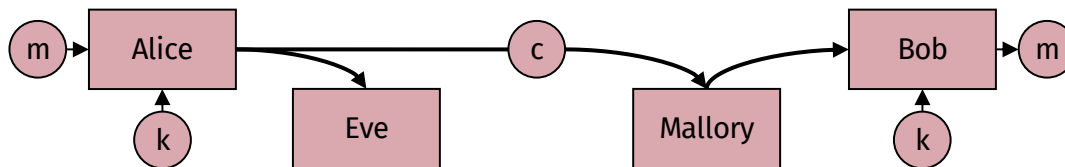


Asymmetric Cryptography (public key cryptography)



Terminology for Encryption Algorithms (2)

- Frequently used terminology
 - A, B: Alice, Bob (communication partners)
 - m: Plaintext
 - c: Ciphertext
 - k: Shared secret key
 - pk_{Bob} : Public key of Bob
 - sk_{Bob} : Private key of Bob
 - E: Eve (passive attacker)
 - M: Mallory (active attacker)
 - T: Trent (trusted third party)



Symmetric Encryption



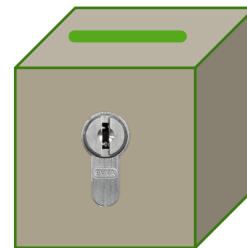
Asymmetric Encryption



Alice



Bob



Public key



Private key

Bitcoin Identities

- Alice can create a keypair. By proving knowledge of her secret key, she can prove her identity (→ Digital Signatures)
- Identities in Bitcoin are pseudonymous
- Alice can create multiple pseudonymous identities

Transactions

- In its simplest form, Alice should be able to transfer a well defined amount of cryptocurrency to Bob.



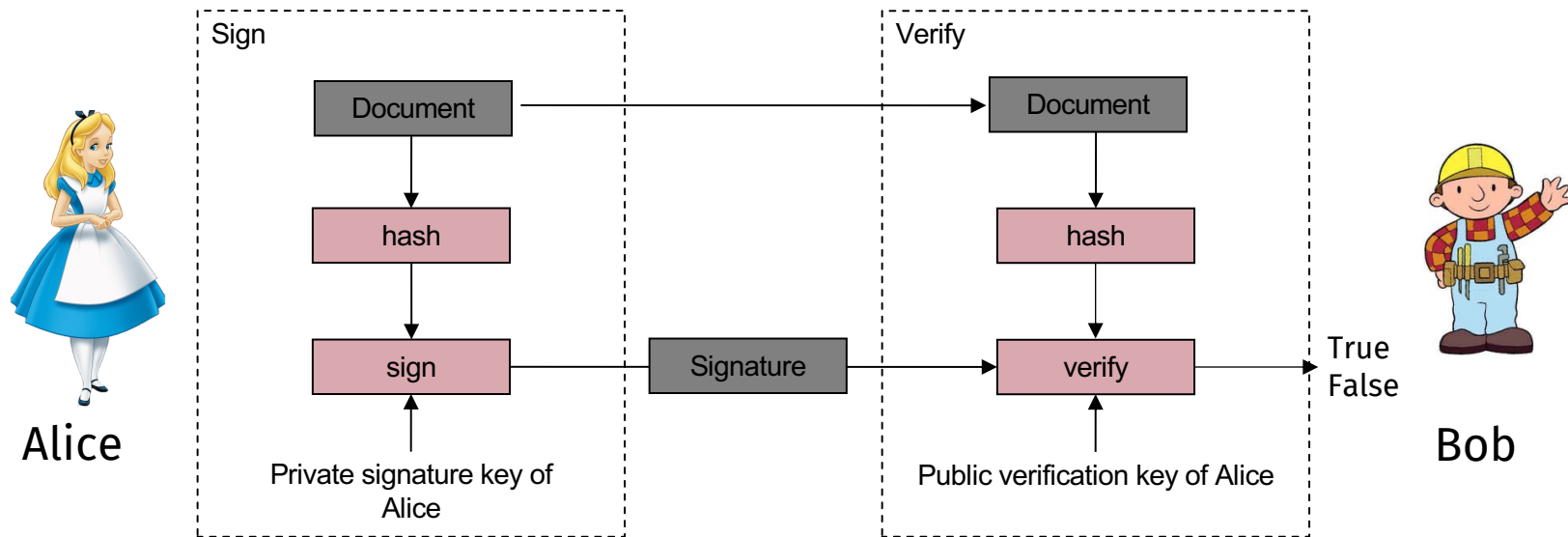
Alice



Bob

- Questions:
 - How does one know that transaction is really from Alice? → Digital Signatures
 - How does one know that Alice actually owns 5฿? → UTXO Model

Digital Signatures



- To reduce signature size and increase speed: hash document first (create “checksum” of document)

Digital Signatures (2)

- Digital Signatures provide
 - **Message Origin:** Signature was created by owner of private key (corresponding to public key used for verification)
 - **Non-repudiation:** creator of signature cannot later deny to have created it
 - **Message integrity:** if message gets altered, signature will become invalid
- Bitcoin uses ECDSA Algorithm with 256 bit keys
 - Algorithm not shown here

Keys and Addresses in Bitcoin

- Private key $sk_{\text{Alice}} = \text{random number } n$
- Public key $pk_{\text{Alice}} = n * P \pmod{p}$
- Computational discrete logarithm problem: getting sk from pk
 - For certain curves, this is practically infeasible for sufficiently large numbers
- $\text{PubKeyHash}_{\text{Alice}} = \text{RIPEMD160}(\text{SHA256}(pk_{\text{Alice}}))$
 - RIPEMD160 and SHA256 two cryptographic hash functions
- $\text{Bitcoin_Address}_{\text{Alice}} = \text{Base58CheckEncode}(\text{PubKeyHash}_{\text{Alice}})$
- Base58CheckEncode includes
 - 1 byte version number
 - Base58Encode translates binary number into text string using only digits and A-Z / a-z except I l 0 O (ambiguous)
 - 4 byte checksum

Cryptographic Hash Functions (1)

- Definition hash function (scatter value function):
 $h: D \rightarrow S$, with $|D| \geq |S|$
 - Desired properties:
 - **Compression:** $|D| \gg |S|$
 - **Chaotic:** slightest change in D (1 bit) should result in a maximal change in S (bit flip with 50% probability)
 - **Surjective:** $|S|$ is fully used
 - **Efficient:** h can be calculated even on large inputs quickly
 - Example for hash function: CRC-32
- with these properties not yet a cryptographic hash function!

Cryptographic Hash Functions (2)

- Cryptographic hash function
 - **One-way function (first pre-image resistance)**
given h with $\text{hash}(m) = h$, then m cannot be calculated efficiently
 - **Weak collision resistance (second pre-image resistance)**
given h with $\text{hash}(m) = h$, no m' can be found efficiently where $\text{hash}(m') = h$ and $m \neq m'$
 - **(Strong) Collision resistance**
no two different m and m' in D with $\text{hash}(m) = \text{hash}(m')$ can be found efficiently
- Examples: SHA-3, SHA-2
- Outdated: SHA-1, MD5

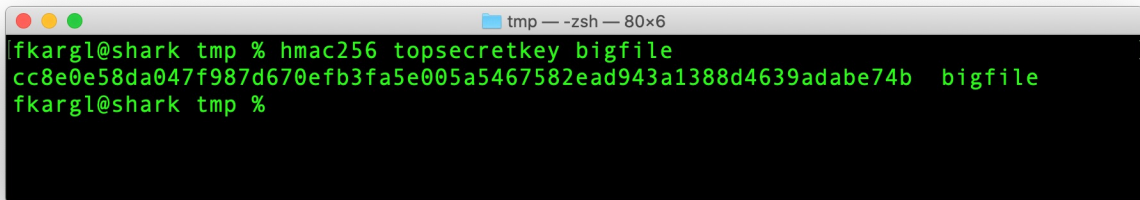
Example

```
fkargl@shark ~ % ls -lh bigfile
-rw-r--r-- 1 fkargl staff 1,0M 3 Okt 14:52 bigfile
fkargl@shark ~ % shasum -a 256 bigfile
d1206e1aa0f4c77547b91a9f9cd87bd96c6306dafbade2d3da1c185e0165da53 bigfile
fkargl@shark ~ % echo "a" >>bigfile
fkargl@shark ~ % shasum -a 256 bigfile
46e6a4e6de91e14aceb5cfa2e6bfec59aa9fe4cdb58e5d3b16a8efb02fab2f6 bigfile
fkargl@shark ~ % time shasum -a 256 bigfile
46e6a4e6de91e14aceb5cfa2e6bfec59aa9fe4cdb58e5d3b16a8efb02fab2f6 bigfile
shasum -a 256 bigfile 0,02s user 0,01s system 93% cpu 0,032 total
fkargl@shark ~ %
```

- Compression: 1MB → 256 bit
- Chaotic: add or change 1 Byte → totally different value
- Efficient: compute hash value on 1 MB file in 1/10 s
- One-way function: only knowing hash, can't reconstruct file
- Weak collision resistant: can't find a second file with hash value d1206e1aa0f4c77547b91...
- Strong collision resistant: can't find any two (different) files with same hash value

Cryptographic Hash Functions (3)

- Message Authentication Code (MAC), Message Integrity Code (MIC), or Keyed Hash
 - Hash function h for integrity assurance of x using key k : $h_k(x) = y$
- HMAC \approx Digital signature with symmetric cryptography
 - whoever knows k and y can check whether x has changed
- Example of Hash-based MAC (HMAC):
 - $\text{HMAC}_k(x) = h([k \oplus p1] || h([k \oplus p2] || x))$
 - h : any cryptographic hash function
 - p_x : predefined Padding values
 - \oplus is XOR, $||$ is concatenation
- Example:
HMAC-SHA256:
HMAC using SHA256 as hash function

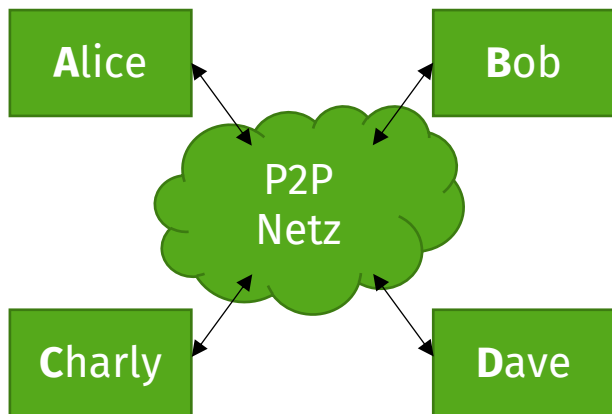
A terminal window titled 'tmp - zsh - 80x6' showing a command and its output. The command is 'hmac256 topsecretkey bigfile' and the output is a long hexadecimal string followed by the filename 'bigfile'.

```
fkargl@shark tmp % hmac256 topsecretkey bigfile
cc8e0e58da047f987d670efb3fa5e005a5467582ead943a1388d4639adabe74b  bigfile
fkargl@shark tmp %
```


UTXO Model

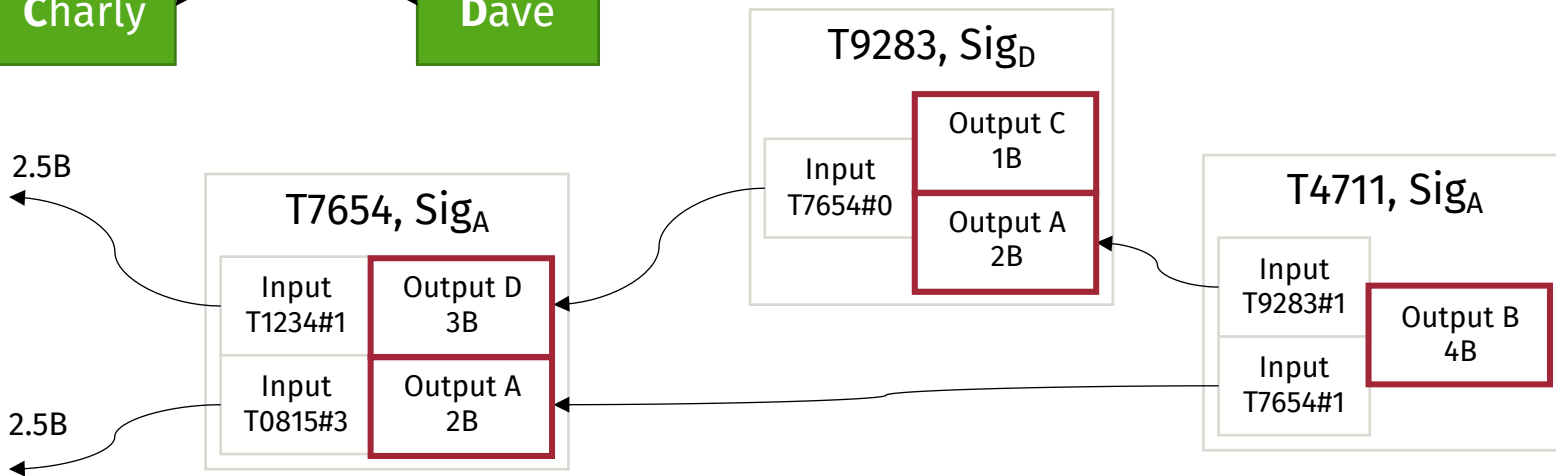
- UTXO: Unspent Transaction Outputs
- How can Alice proof to Bob that she owns the 5 Bitcoins?
 - Manage an account balance; or
 - Refer outputs to inputs from previous transactions (UTXO)
- Discuss the pros and cons of each approach

UTXO Example



UTXO

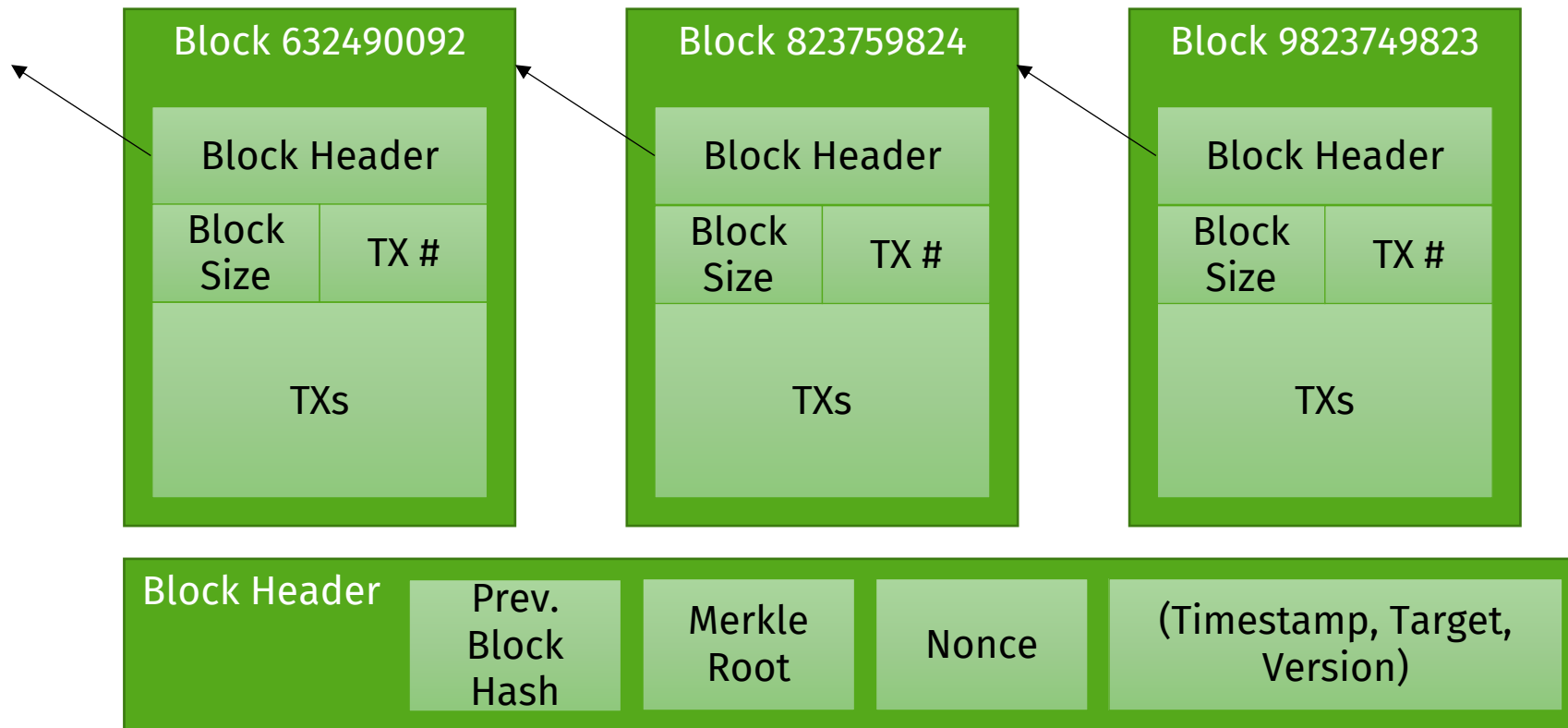
- Amount of ₿ in system stays constant
- Everyone needs to know all transactions to be able to determine if output is still unspent



Distributed Database

- Also called Distributed Ledger
 - Distributed Ledger Technologies (DLTs)
- Transactions are broadcasted to all participating miners in a Peer-to-Peer Network
 - P2P Networks would be a lecture on its own, so just accept that there is a magic network so that all transactions and later blocks will be magically seen by everyone eventually, but not necessarily at the same time and in the same order
- Agreement on order is important, otherwise double spending cannot be decided
- Agreeing on order is costly
 - Group unrelated transactions in blocks
 - Agree on order of blocks only
 - Therefore Blockchain

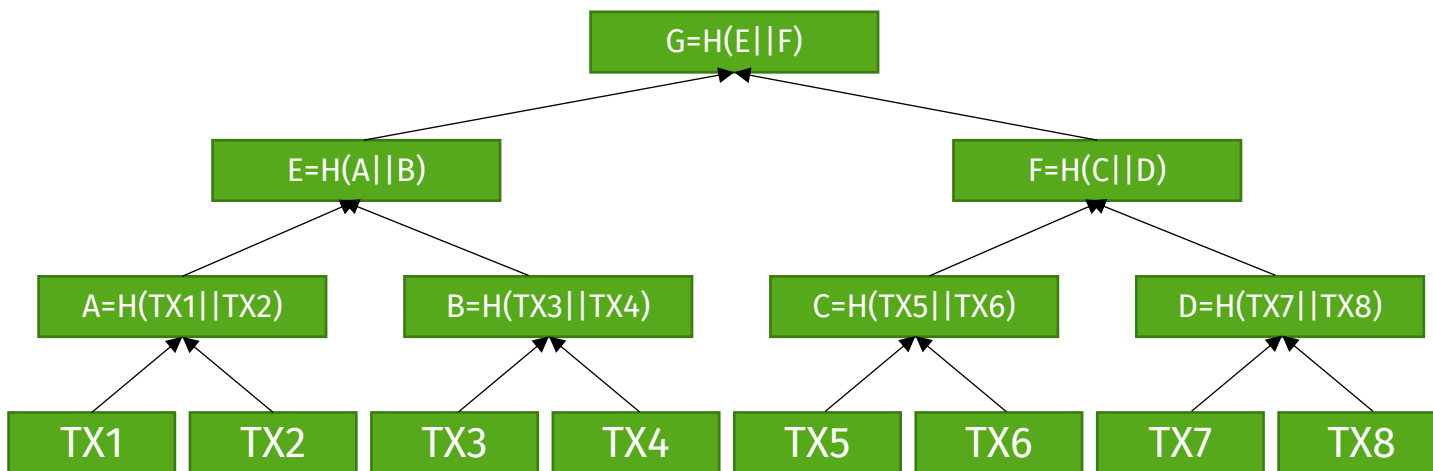
The Blockchain



Block-ID

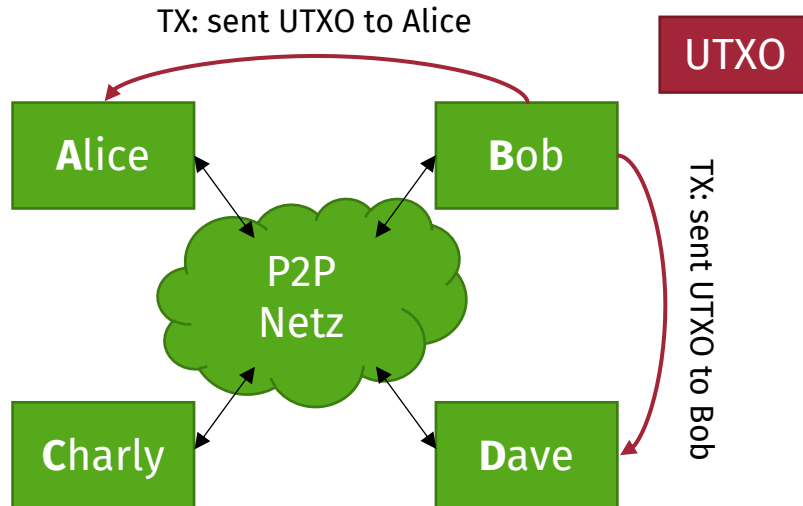
- Block-ID = $H(\text{blockHeader}) = H(\text{prevBlockHash} || \text{Merkle Root} || \text{Nonce} || \dots)$
- Changing anything inside the block, changes the Block-ID
- Changing anything inside a previous block, invalidates the whole blockchain from there on
- What is a Merkle Tree and a Merkle root?

- G: Merkle Root
- Any change in any TX will lead to new G
- For n transactions, tree has height of $\log_2(n)$
- $O(\log_2(n))$ hash operations to build tree
- Proof of inclusion: Providing TX1, B, F, and G can serve as a proof that TX1 is included in the tree
- Due to nature of hash functions, nobody can build a different tree leading to same root



Consensus

- So how can everyone agree on valid TXs, blocks, and order of blocks without central entity?
- What if we have two transactions spending same UTXO? Which one is valid and which one not?
- Ideas (other than PoW)?



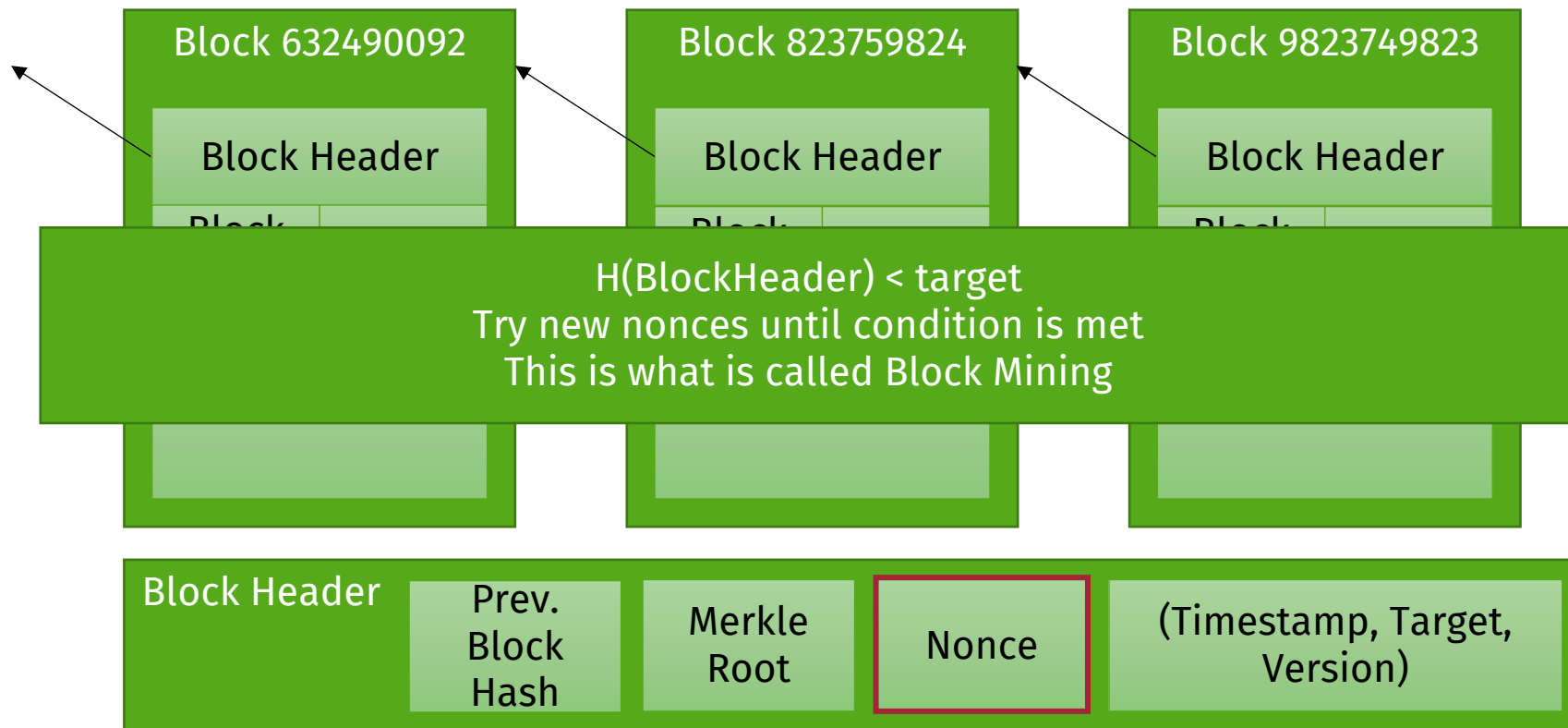
Consensus (2)

- No problem, Computer Science has you covered!
- Nodes can vote on correct TXs and reject incorrect ones
- PBFT Protocols (Practical Byzantine Fault Tolerance)
 - Achieve Consensus in the presence of Faulty or Malicious Nodes
 - Can tolerate up to $f < (n - 1) / 3$ faulty nodes (n: total nodes in system)
 - What's the problem?
- What is n in Bitcoin?
- What about Bob just creating $f+1$ identities and voting with them?
- Doesn't seem to work for Bitcoin (but we come back to this in a future lecture!)

Proof-of-Work

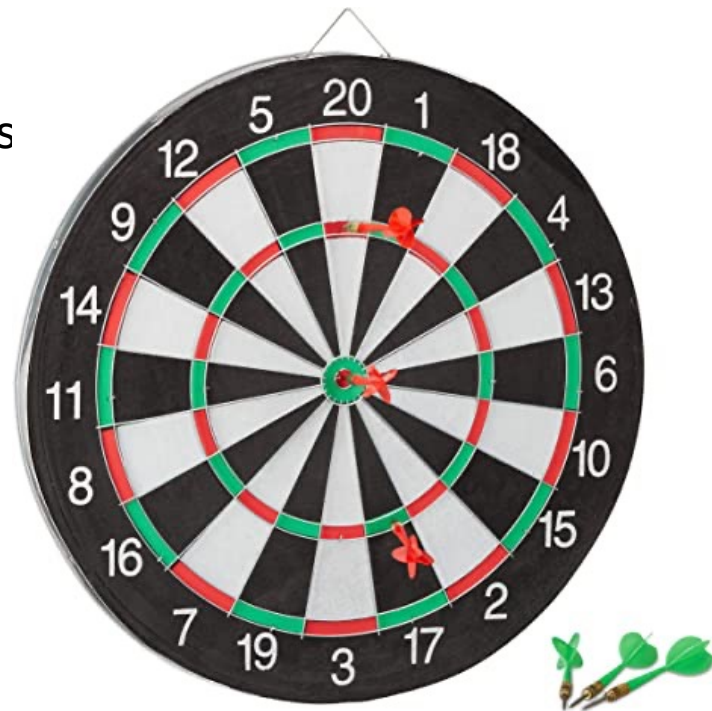
- Proof-of-Work: Limit voting capacity by resource usage (Proof-of-Work)
 - Computers have to solve a compute-intensive puzzle based on proposed new block
 - While Bob can replicate its identities, it cannot increase its compute power arbitrarily
 - Solution of the puzzle is considered proof-of-work and makes a block valid
 - Only valid blocks are considered as valid extensions of the blockchain
 - Forking: still two conflicting blocks may be proposed in parallel
 - As few blocks appear and conflicts are rare, further consensus on correct extension of blockchain can be achieved by just accepting longest chain as valid
- Bitcoins Partial Preimage Hash Puzzle: $H(\text{BlockHeader}) < \text{target}$
 - Computational difficult (hashfunctions are chaotic!)
 - Parameterizable (target calculated such that on average one block every 10 min found, independent of hash-rate)
 - Easily verifiable (just check the condition)

Blockchain Mining



Mining Intuition

- It's like blindfoldedly throwing darts at a target
- Size of inner ring determines difficulty
- Rate (darts/s) at which you throw darts determines success rate (mining rate)
- More miners should not lead to higher rate of valid blocks being found
- Target determines leading number of '0's of hash
- Difficulty algorithmically chosen such that on average 2016 blocks are created per 2 weeks or one every 10 minutes
 - $\text{difficulty} = \frac{\text{two_weeks}}{\text{time_to_mine_prev_2016_blocks}}$



Mining Intuition

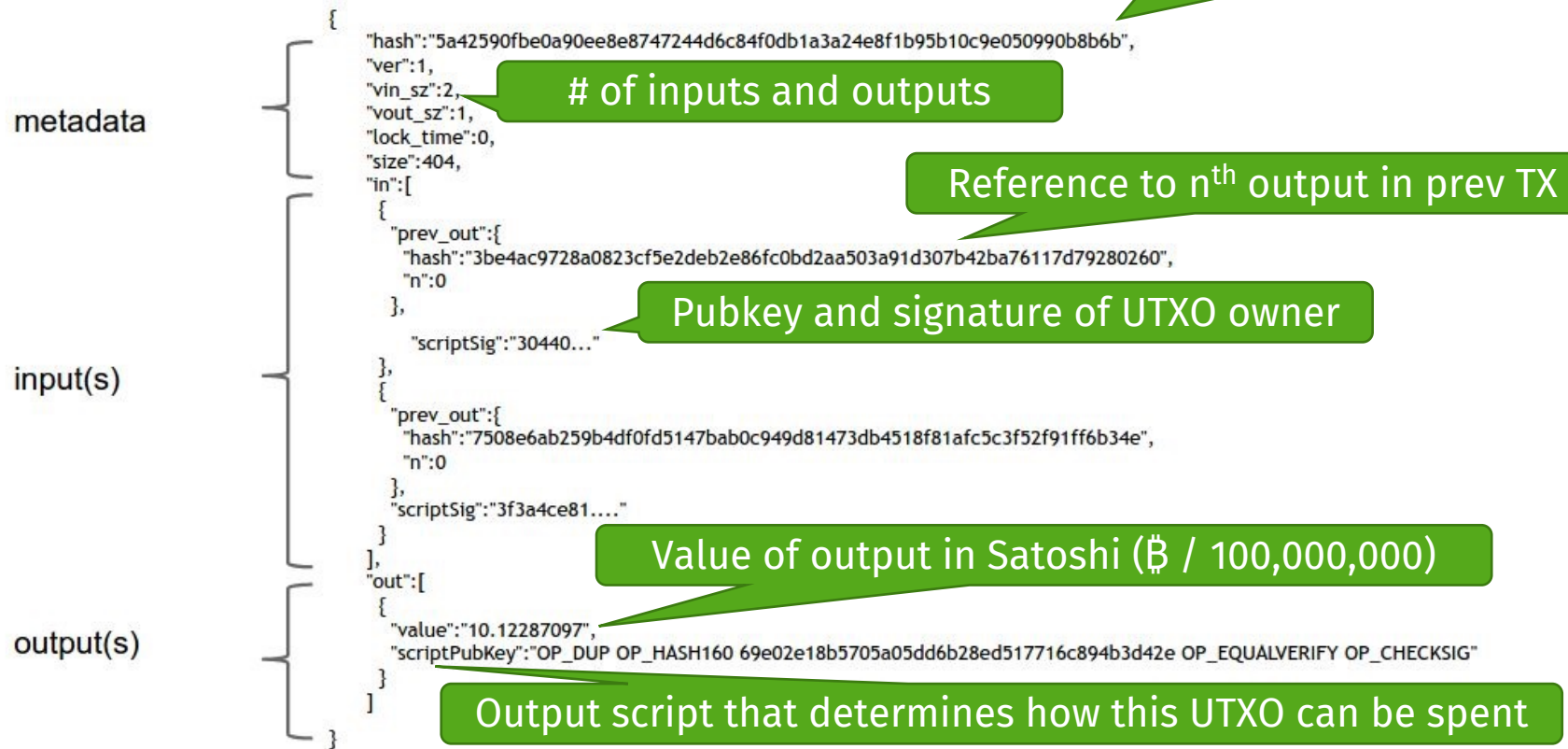
- *difficulty* algorithmically chosen such that on average 2016 blocks are created per 2 weeks or one every 10 minutes
 - $\text{difficulty} *= \text{two_weeks} / \text{time_to_mine_prev_2016_blocks}$
- What if old *difficulty* = 10 and
 - $\text{time_to_mine_prev_2016_blocks} = 1 \text{ week?}$
 - $\text{time_to_mine_prev_2016_blocks} = 4 \text{ weeks?}$



Bitcoin Script

- Bitcoin transactions are expressed in a stack-based scripting language
- This allows to express from simple to more complex transaction types
 - but misses some elements from classical computer languages like loops
- Transactions map UTXO inputs to outputs
- Let's look at an example

Bitcoin Transaction Example



Bitcoin Transaction Example (2)

- Locking Script (as it locks a UTXO)
 - Locks UTXO and allows it to be spend by someone who can create a matching unlocking script
 - Example: scriptPubKey → need a matching signature for unlocking
- Unlocking Script (as it spends an UTXO)
 - Allows to use a locked UTXO as an input
 - Example: scriptSig (matches scriptPubKey)

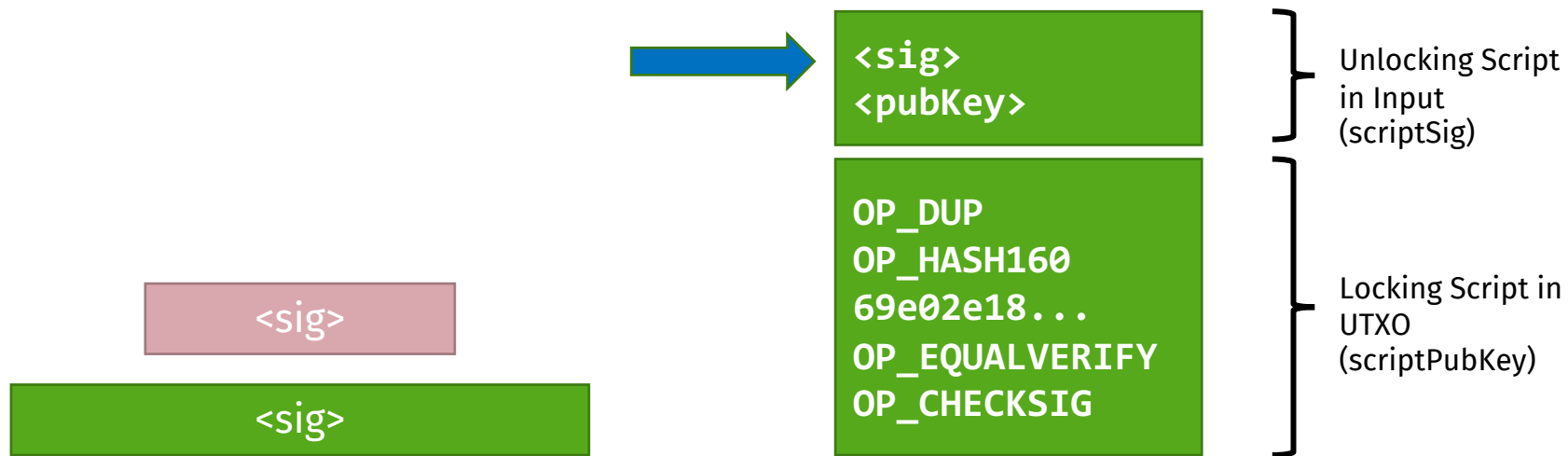
<sig>
<pubKey>

} Unlocking Script
in Input
(scriptSig)

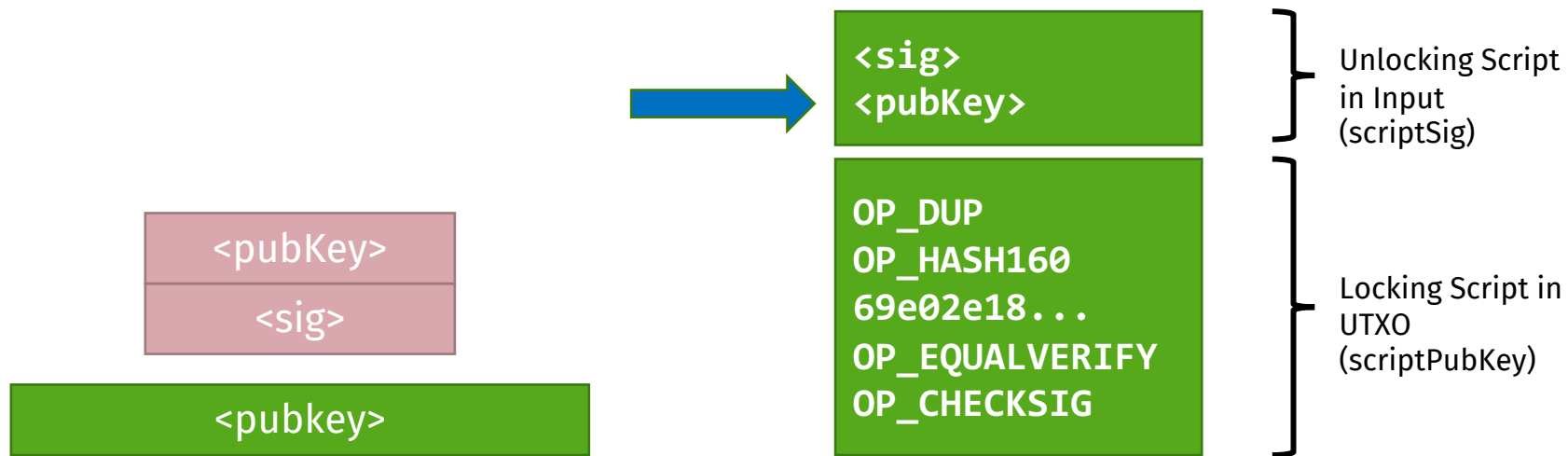
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY
OP_CHECKSIG

} Locking Script in
UTXO
(scriptPubKey)

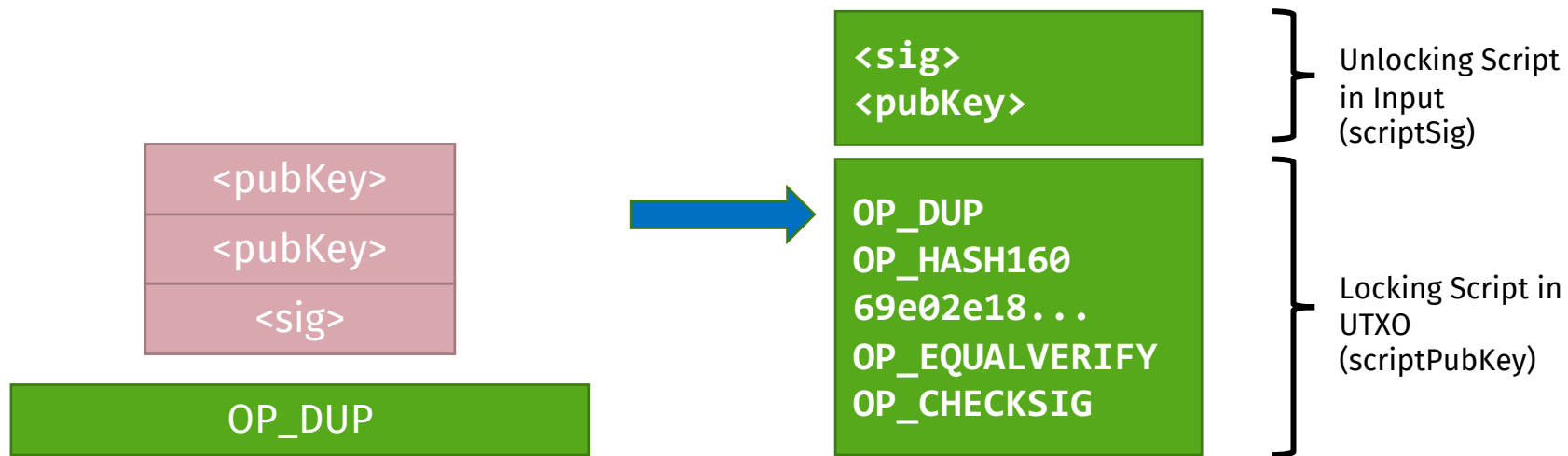
Bitcoin Transaction Example (3)



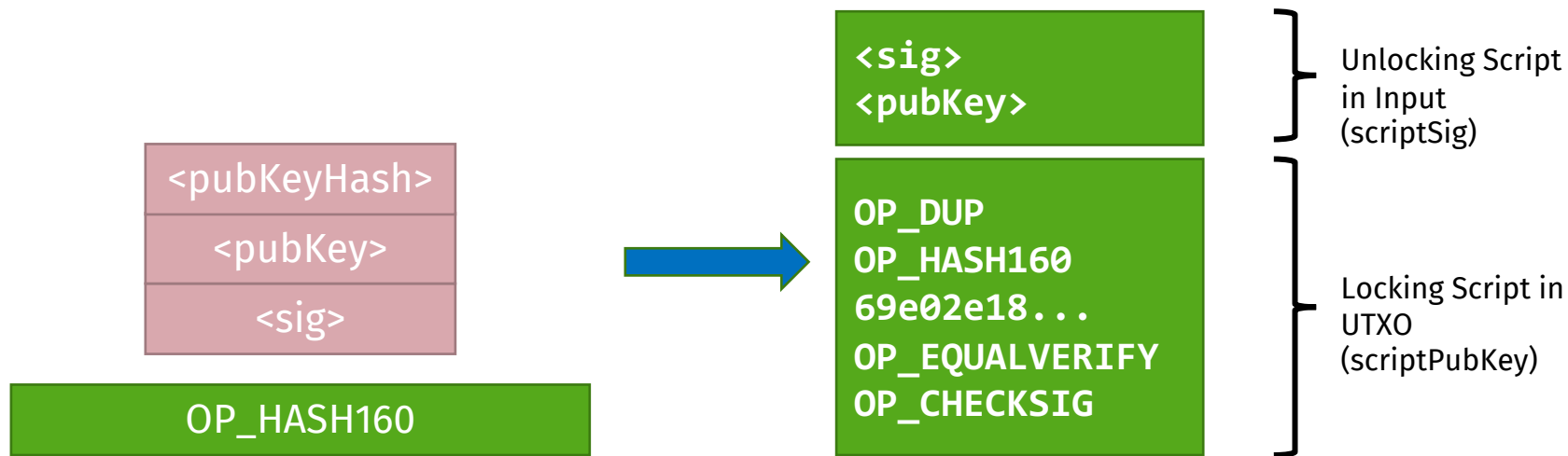
Bitcoin Transaction Example (4)



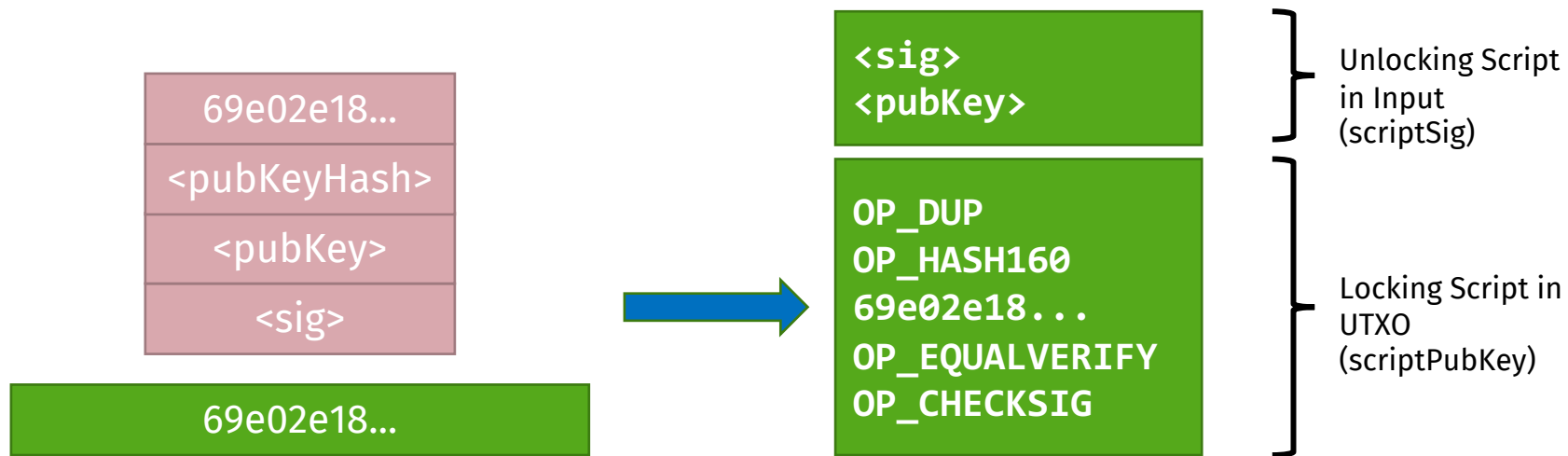
Bitcoin Transaction Example (5)



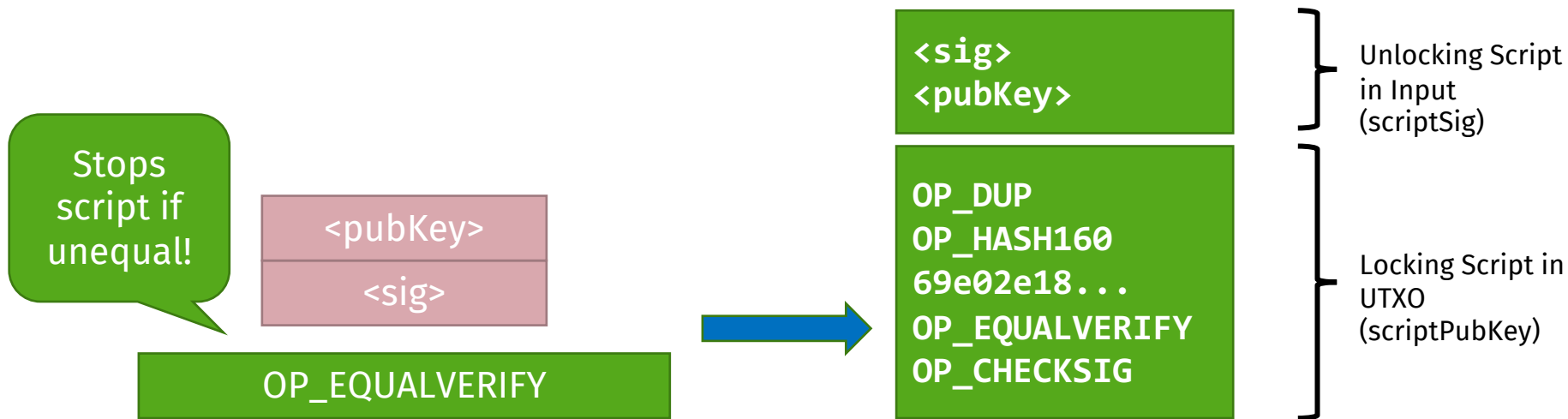
Bitcoin Transaction Example (6)



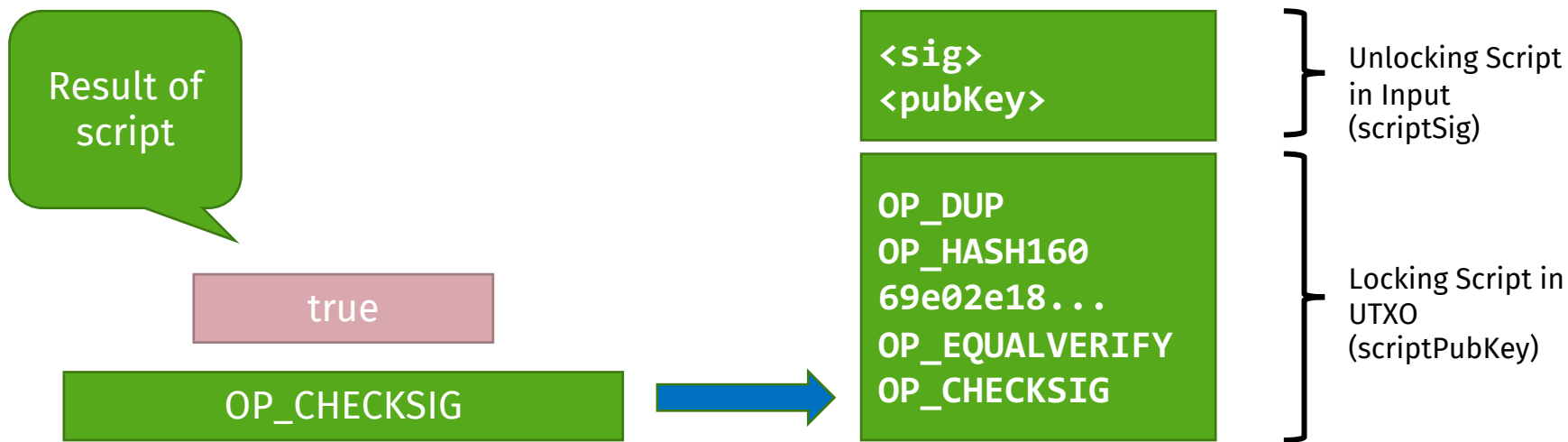
Bitcoin Transaction Example (7)




Bitcoin Transaction Example (8)



Bitcoin Transaction Example (9)



Blockchain.info Demo: Block 50,001





Bitcoin Block #50,001

Mined on 4/10/2010, 18:31:40 [View all blocks](#)

This block was mined on 4/10/2010, 18:31:40 by **Unknown**. A total of 150.00 BTC (\$0.00) were sent in the block with the average transaction being 50.00000 BTC (\$0.00). Unknown earned a total reward of 50.00 BTC \$0.00. The reward consisted of a base reward of 50.00 BTC \$0.00 with an additional 0.00000 BTC (\$0.00) reward paid as fees of the 3 transactions which were included in the block.

← →

Details	
Hash	00000-ee141 
Depth	709,981
Capacity	0.06%
Distance	12y 6m 14d 22h 20m 58s
BTC	150.0000
Value	\$0.00
Value Today	\$2,878,107
Average Value	50.000000000000 BTC
Median Value	100.000000000000 BTC
Input Value	150.00 BTC
Output Value	200.00 BTC
Transactions	3
Witness Tx's	0
Inputs	4
Outputs	3
Size	647
Version	0x1
Merkle Root	ee-49 
Difficulty	6.09
Nonce	56,717,043
Bits	472,518,933
Weight	2,588 WU
Median Time	10. Apr. 2010, 18:31:40
Mined on	10. Apr. 2010, 18:31:40
Height	50,001
Confirmations	709,981
Miner	Unknown
Coinbase	L

Transactions

↕ Last First ↗ Value ↘ Value ↗ Fee

↘ Fee

TX 0 • Hash **e188-4d58**
4/10/2010, 18:31:40

50.000000000 BTC \$959,369
Fee 0 Sats \$0.00

TX 1 • Hash **7940-455a**
4/10/2010, 18:31:40


100.000000000 BTC \$1,918,738
Fee 0 Sats \$0.00

TX 2 • Hash **f847-c60a**
4/10/2010, 18:31:40

50.000000000 BTC \$959,369
Fee 0 Sats \$0.00




TX 0 • Hash **e188-4d58**
4/10/2010, 18:31:40

50.000000000 BTC \$959,369
Fee 0 Sats \$0.00

From	To
Coinbase	1. 1PkqKGbNL-FH77ADrPQ  50.000000000 BTC Scripts

TX 1 • Hash **7940-455a**
4/10/2010, 18:31:40

100.000000000 BTC \$1,918,738
Fee 0 Sats \$0.00

From	To
1. 1DoQeKfU3-fCPDqSPae  50.000000000 BTC Scripts	1. 1HaHTfmvo-4cHNmBQHQ  100.000000000 BTC Scripts
2. 1PZMgkF2x-3SjnQE1m9  50.000000000 BTC Scripts	

1DoQeKfU3-fCPDqSPae**Pkscript**

```
047d78caf51bd9ed289bb3203eb1baa8585df39fabe76ad51107ad6741d8c
d775ea46f750af5931335a58ee8fab669b21abc7c296367229750e852d53d
de6b5e7
OP_CHECKSIG
```

Sigscript

```
493046022100e0bc99e312e428ea1559b5bc2e3210f3a7202a7f8b2ee124c
6ea9aeda497ecac022100ccda2dc6aac965b4ba3116bc529838538ac0c99f
e1a295941d00b5e351f86f0f01
```

Witness**1HaHTfmvo-4cHNmBQHQ****Pkscript**

```
OP_DUP
OP_HASH160
b5cd7aaed869cd5ccb45868e8666e7e934a23736
OP_EQUALVERIFY
OP_CHECKSIG
```

More Aspects of Bitcoin

- More things to do with Bitcoin script: P2SH Locking and Multisig, Timelocks, ...
- Bitcoin in real life:
 - wallets, different types of wallets, simple payment verification, and miners
- Bitcoin Governance (maintaining and changing the bitcoin protocol and software, hard-/soft-fork)
- Attacks on Bitcoin (51% attack, ...), game theoretic perspective
- Bitcoin confidentiality / privacy



designed by Startline - Freepik.com

L03-05: Ethereum

Frank Kargl | Blockchain Fundamentals | 2023-10-30 ff



Institut für Verteilte Systeme
Institute of Distributed Systems

Smart Contracts and Ethereum

- Bitcoins advantages:
 - Pseudonymous cryptographic identities
 - Decentralized trust through consensus protocol (democratic)
 - Immutable ledger of truth (tamper-proof immutable ledger)
 - Uncensorable (cannot be controlled by any single party)
 - Distributed (single point of failure)
- BUT:
 - Application limited by transaction-oriented script language
- Can we design a system similar to Bitcoin but with much higher degree of flexibility?

Smart Contract Example



Pay 50.000 \$ to farmer,
iff

- Farmer paid 1000 \$
fee into contract, and
- Temperature in
October is below 0°
Celsius

"[Dieses Foto](#)" von International Maize and Wheat ... ist lizenziert gemäß [CC BY-SA-NC](#)

"[Dieses Foto](#)" von Florian Hardwig ist lizenziert gemäß [CC BY-SA-NC](#)



Ethereum

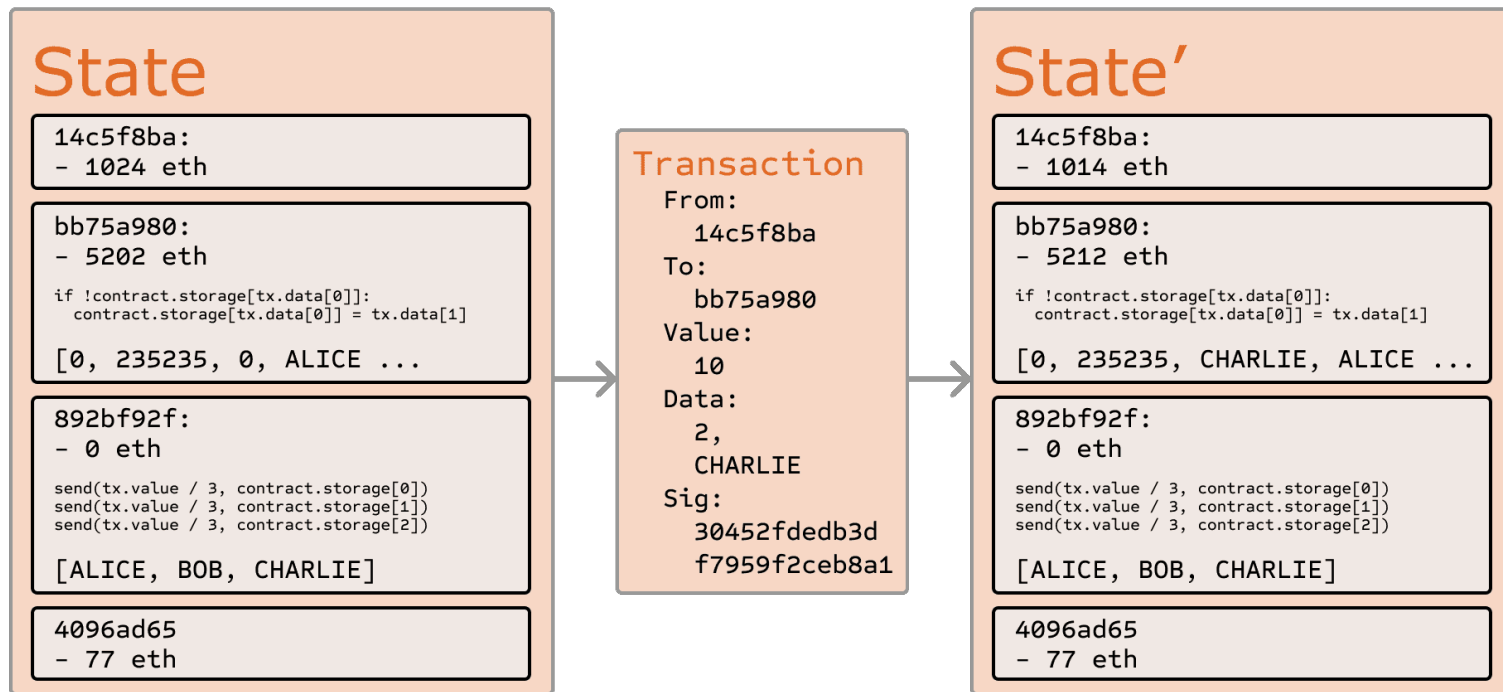
- Ethereum retains a lot of the design of Bitcoin, but
 - is a decentralized platform to run smart contracts
 - Account-based blockchain (no UTXO, but account has balance and state)
 - External accounts (“users”)
 - Contract accounts
 - Includes a native asset / currency called Ether
 - Proof-of-Stake-based mining process
 - Used Proof-of-Work until 2021 very similar to Bitcoin
 - See later for PoS
 - Nodes run EVM to validate transactions



Ethereum Smart Contracts

- Turing-complete programming language implemented in Ethereum Virtual Machine (EVM)
 - Multiple higher-level programming languages like Solidity or Vyper compile to EVM code
- Like a distributed computer where all nodes jointly execute the same programs to achieve the same results
- Distributed state-machine where transactions change the global state in a synchronized way
 - Transaction == state transition function
- Smart contracts are like autonomous agents:
 - Live inside the Ethereum network
 - React to external event when triggered by a transaction
 - Control
 - Internal ether balance
 - Internal contract state (“variables”)
 - Permanent storage

Ethereum State Transitions



Source: Ethereum Whitepaper, ethereum.org

Challenges in EVM Code

- Indeterminism?
 - What if we have `x=rand()` in our code???
 - Solution: no indeterminism
- Loops?
 - `while(true) { complex_calculation(); }`
 - Halting problem: Cannot be determined by compiler or pre-runtime check
 - Denial-of-Service Attack
 - Solution: Transaction fees in gas
 - Transactions specify the initial gas supply (startgas) and the maximum price they will pay for gas (gasprice)
 - `startgas * gasprice` is subtracted from sender's ether balance
 - If transaction successfully executes: remaining gas is refunded
 - If transaction runs out of gas: it is reverted

Solidity Example

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;
contract Counter {
    uint public count;

    // Function to get the current count
    function get() public view returns (uint) {
        return count;
    }

    // Function to increment count by 1
    function inc() public {
        count += 1;
    }

    // Function to decrement count by 1
    function dec() public {
        // This function will fail if count = 0
        count -= 1; }
}
```

<https://solidity-by-example.org/first-app/>

Use Cases for Smart Contracts?

- Brainstorm for 5 min with your neighbours on reasonable use-cases for Smart Contracts

Alternative Consensus Mechanisms

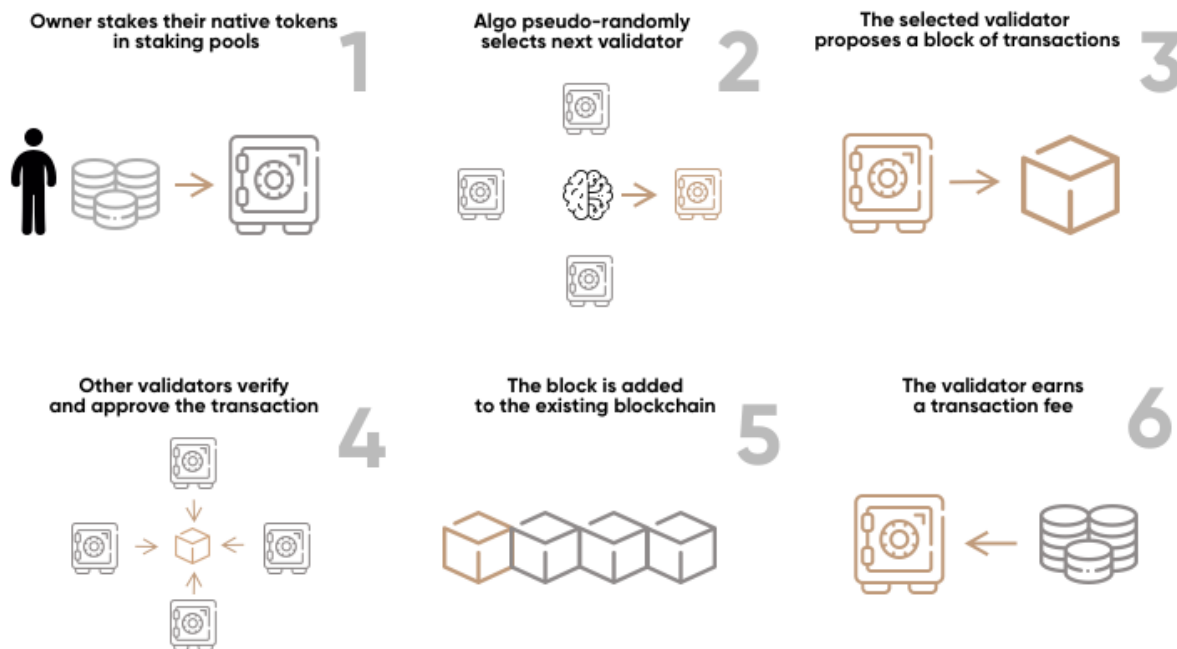
- What are the drawbacks of Proof-of-Work?
 - Wastes a lot of energy and hardware for no real purpose
 - Leans towards centralized mining farms (not “One CPU one Vote”)

Alternative Consensus Mechanisms

- Other consensus mechanisms:
 - Proof-of-Useful-Work (e.g., protein folding, SETI, search for large primes,...)
 - Problems: no inexhaustible problem space, no equiprobable solution space, no decentralized algorithmically generated problem
 - Proof-of-Human-Work (Kopp, Kargl, Bösch, Peter, “uMine: A Blockchain Based on Human Miners.” In: ICICS 2018. LNCS 11149. 10.1007/978-3-030-01950-1_2)
 - Proof-of-Stake as in Ethereum

Proof-of-Stake in Ethereum

HOW STAKING WORKS IN THE PROOF-OF-STAKE CONSENSUS MECHANISM

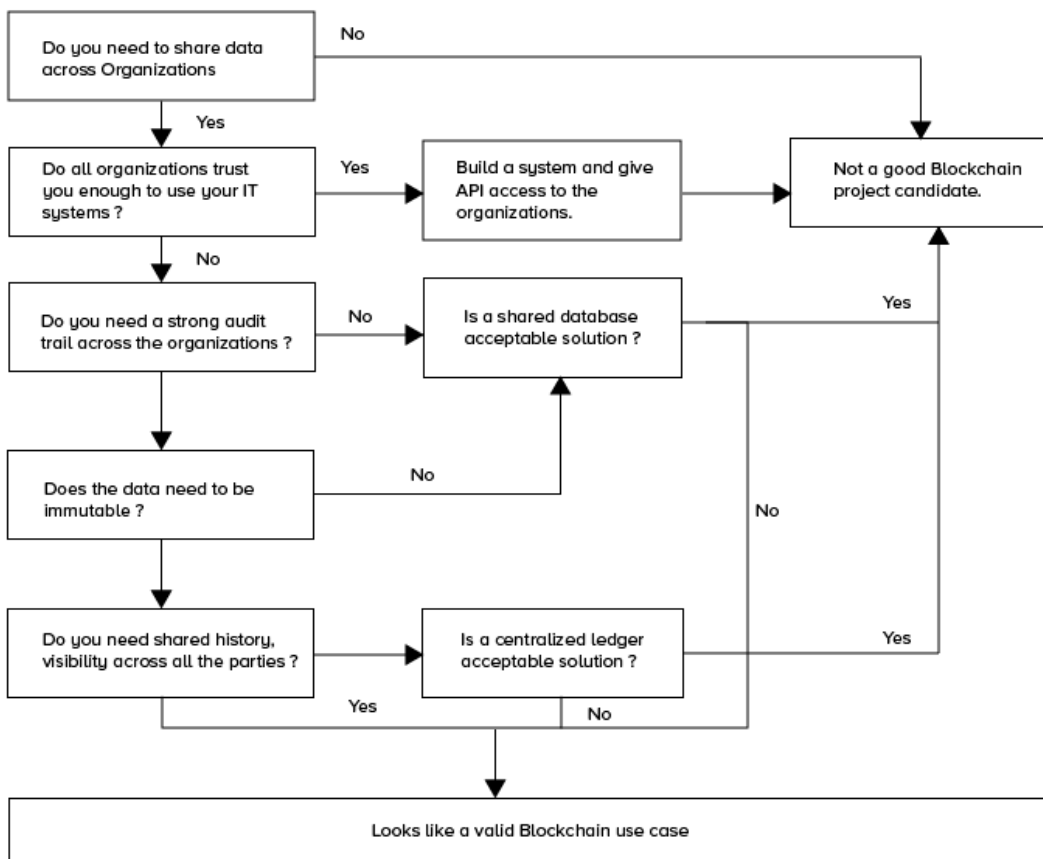


Source: SEBA Research

Enterprise Blockchains

- Use inside enterprises or consortia with limited and authorized participants allows use of PoS and PBFT-Protocols
- Frequently used platform: Hyperledger Fabric
- Note: trust models tend to become more and more centralized like distributed transaction frameworks, Blockchain not really needed

Enterprise Blockchains



From:
<https://appinventiv.com/blog/choose-best-blockchain-development-platform/>



designed by Startline - Freepik.com

L03-05: Blockchain & Privacy

Frank Kargl | Blockchain Fundamentals | 2023-10-30ff



Institut für Verteilte Systeme
Institute of Distributed Systems

Terminology

Confidentiality:

Only authorized entities have access to protected information.

Information Privacy:

Information privacy is the relationship between the collection and dissemination of data, technology, the public expectation of privacy, contextual information norms, and the legal and political issues surrounding them. It is also known as data privacy or data protection.

Anonymity:

Impossibility of relating information or actions to a specific person.

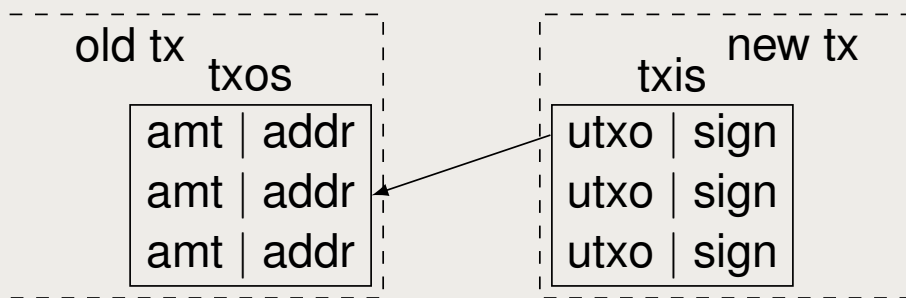
Pseudonymity:

Usage of pseudonymous identifiers to prevent direct identification of persons.

Are Transactions Anonymous?

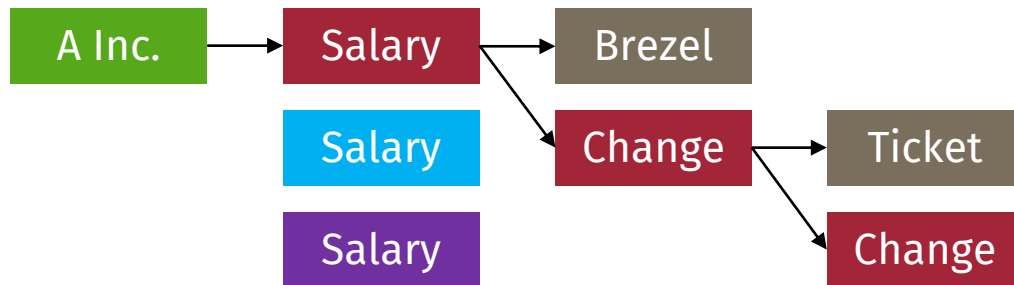
Addresses are only pseudonyms.
Linking them to real persons is feasible and not too difficult.

Graph of transactions



Example: <https://blockchain.info/address/1BvayiASVCmGmg4WUJmyRHoNevWWo5snqC>

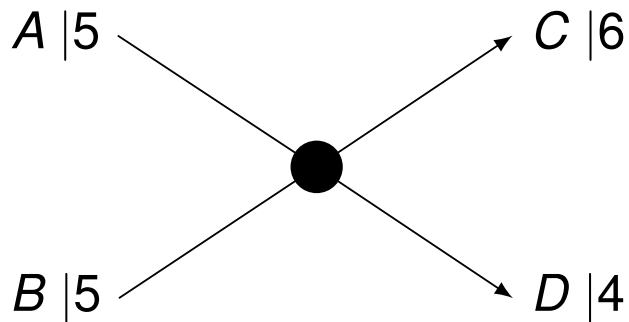
What can be Learned from Transactions?



Airline knows with high probability

- Employed at A Inc.
- Likes Prezels
- Dynamic Pricing

Semantical Interpretation of Transactions



Linking Identities

- C is recipient, D is change
- if D would be recipient, A or B would be enough input

P2P-based Deanonymization

Underlaying peer-to-peer network for broadcast of transactions and blocks

IPs

- log all p2p traffic
- link IP addresses to transactions
- deduce routing / delays etc.

Mitigation

- use Tor for all interactions
- garlic routing with i2p (kovri)

What if Money has a History?

Fungibility

Two coins with the same value are worth the same.

Any Preference?

1 BTC 1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX

1 BTC 1BvayiASVCmGmg4WUJmyRHoNevWWo5snqC

What if Money has a History?

Fungibility

Two coins with the same value are worth the same.

Any Preference?

1 BTC 1F1tAaz5x1HUXrCNLbtMDq...

1 BTC 1BvayiASVCmGmg4WUJmy...

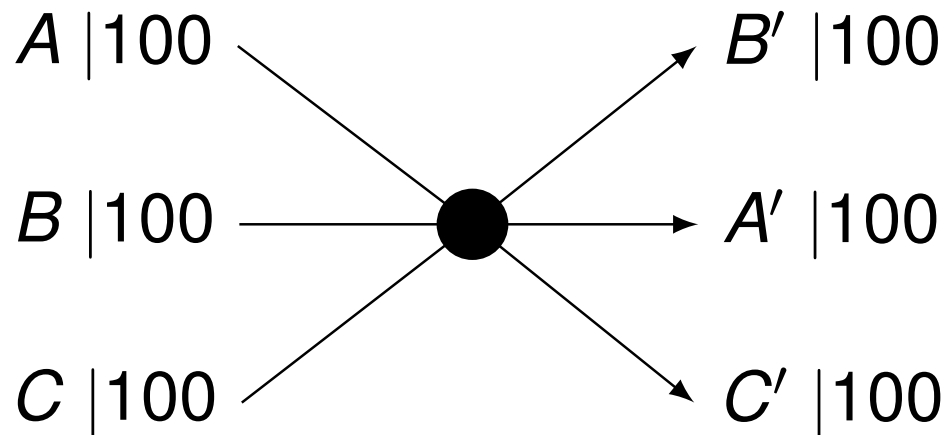
→ FBI seized Silkroad funds

→ bitcoin.de bloggerX

Solutions?

Mixing Service

Try to break link between pseudonym and identity.



Coin Mixing

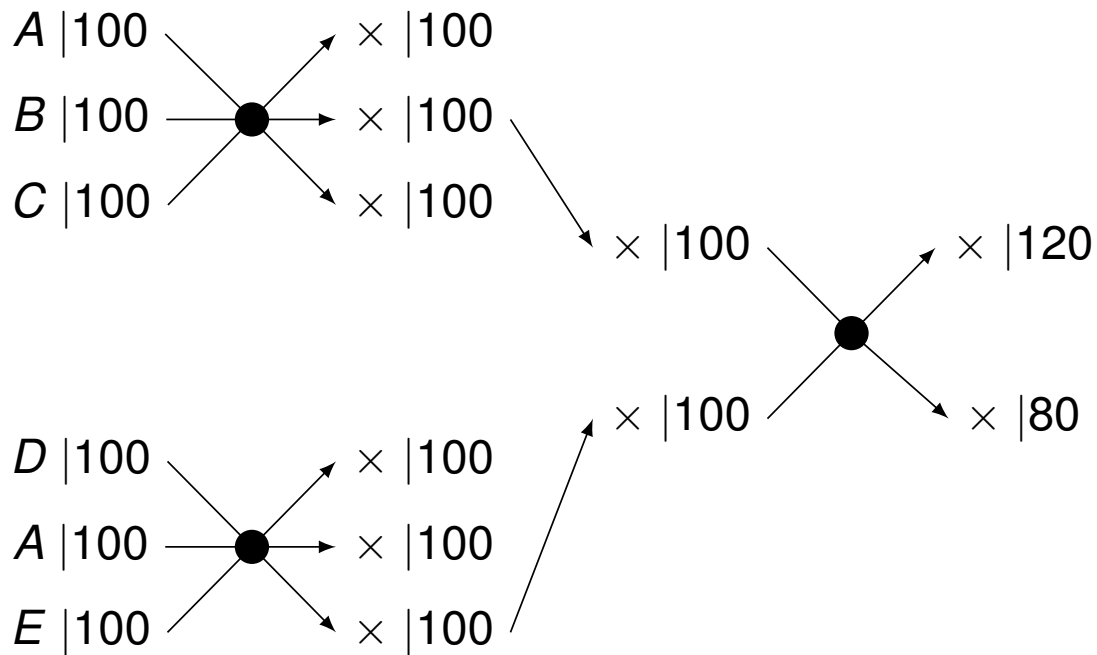
Requirements

- equal amounts
- new set of identities at outputs or reduced utility

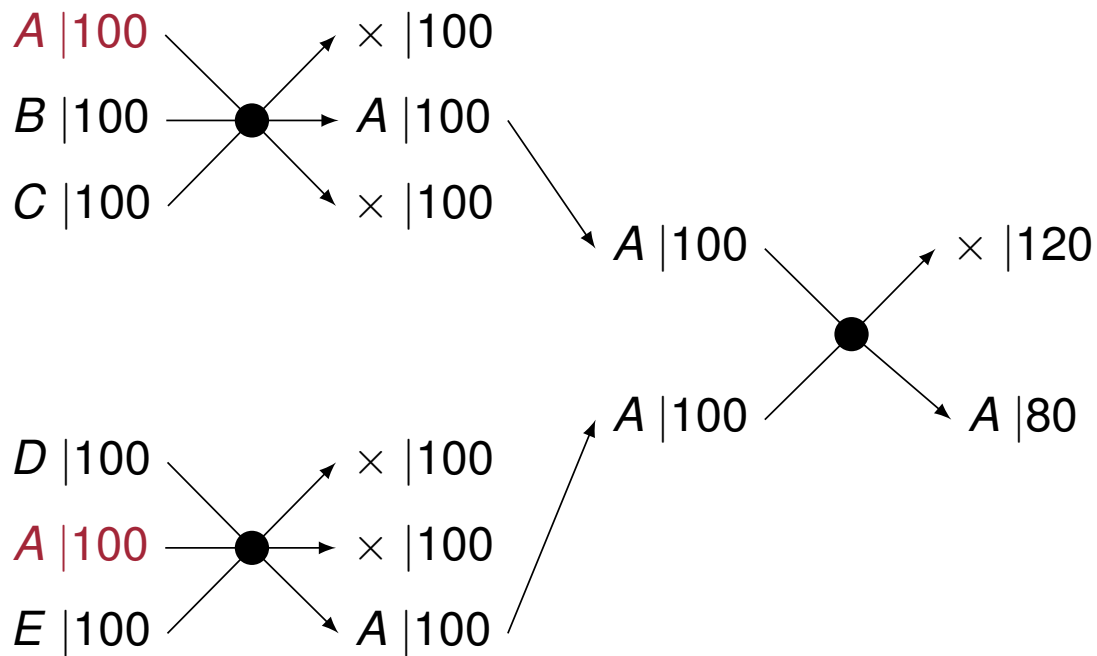
Benefits

- reduce history to probability $1/n \neq 0$

Set Deanonimization



Set Deanonymization



Introduce Anonymization Techniques into Blockchain



Validity of Transactions

Every transaction on a blockchain has to be publicly verifiable by miners

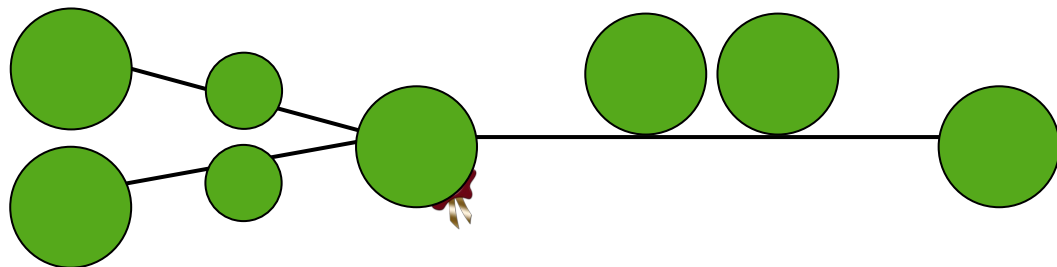
Preservation of Assets

$$\sum A_{in} = \sum A_{out}$$

Proof of UTXO Ownership

$\text{sign}_{sk}(\text{pk})$ for UTXO addresses

Transactions and Privacy Goals



But keep public verifiability!

Validity of Transactions

Every transaction on a blockchain has to be publicly verifiable by miners

Preservation of Assets

$$\sum A_{in} = \sum A_{out}$$

Proof of UTXO Ownership

$\text{sign}_{sk}(\text{pk})$ for UTXO addresses

Ingredients for Confidential Transactions

Receiver Anonymity: One-time Addresses

Sender Anonymity:
Linkable Spontaneous Ad-hoc Group Signature (LSAG)

Amount Confidentiality: Pedersen Commitments

Token Type Confidentiality: Ped. Com. & Boromean Signatures

Receiver Anonymity: One-Time Addresses



Elliptic Curve Crypto in a Nutshell

Curve

- Curve equation, e.g., $-x^2 + y^2 = 1 - (121665/121666) x^2 y^2$
- Base point G

Operations

- $P + P = 2P$, $5P = P + 4P$, $P + Q = R$

Discrete Logarithm

- with $P = xG$, $\log_G(P) = x$ is hard to compute

Receiver Anonymity: One-Time Addresses

- Sender derives a unique new address A' for a recipient A for each transactions
- Only real recipient can recognize that he is intended addressee and spent amount in future TXs

Keys

- secret key (a, b)
- public key $(A, B) = (aG, bG)$
- also partially possible (a, B) (view key)

Receiver Anonymity: One-Time Addresses

Sender

- select r random \rightarrow give r to receiver
- Tx pubkey $R = rG$
- destination key $P = \mathcal{H}(rA)G + B$

Tracking

- Tracking requires (a, B, R)
- $P' = \mathcal{H}(aR)G + B = \mathcal{H}(arG) + B \stackrel{?}{=} P$

Recover

- Recovery requires (a, b, R)
- $P = \mathcal{H}(aR)G + B = (\mathcal{H}(aR) + b)G = p'G \rightarrow$ sign with p'

Resources

- Berkeley Course 198.1X Bitcoin and Cryptocurrencies (freely available online)
 - <https://blockchain.berkeley.edu/courses.html>
- Book by Narayanan et al. “Bitcoin and Cryptocurrency Technologies”,
 - <https://bitcoinbook.cs.princeton.edu>