

Blockchain

Intro

About me

- PhD on Reconfiguration (2016)
- Associate Professor at University of Stavanger Norway (since 2018)
- Last year (2022-2023) at TU Braunschweig Germany



Blockchain

Motivation

Bitcoin

- running > 10 years
- 10.000 nodes
- not possible to buy a coffee



Blockchain

This lectures

Take a look at blockchain technology

- What works
- What does not work

Learning goal: Know how and when to use it.

Know when not to use it.

But: No economics.

Blockchain 1

Hashlist and Merkle trees

Blockchain datastructure

What is a blockchain

A blockchain is an append only log
secured against changed.

Typically a blockchain

- is stored on different nodes

Idea: Log all interactions

- Log enables anyone to reproduce/recreate state.

Cryptographic hash function

Idea

$$H(x) = y$$

- x string or byte array
- y fixed size byte array

looks random: hashing something new gives a random value

is deterministic: hashing something twice gives the same value

Cryptographic hash function

Properties

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo">
</script>
```

... string or byte array / y: fixed size byte array

Properties:

- **Pre-image resistance:**
given y cannot find x s.t. $H(x) = y$
- **Weak collision resistance:**
given x cannot find x' s.t. $H(x) = H(x')$
- **Strong collision resistance:**
cannot find x and x' s.t. $H(x) = H(x')$

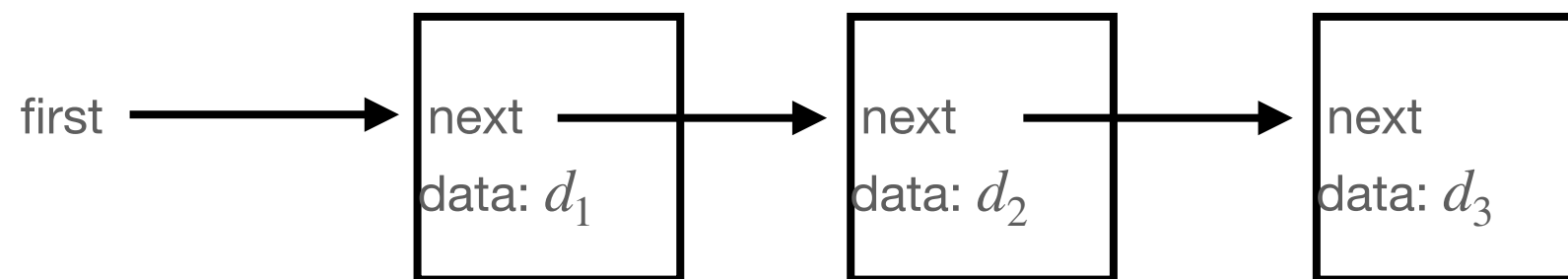
Use cases examples:

- Password hashes
- HTML5 integrity attribute

Hash chain

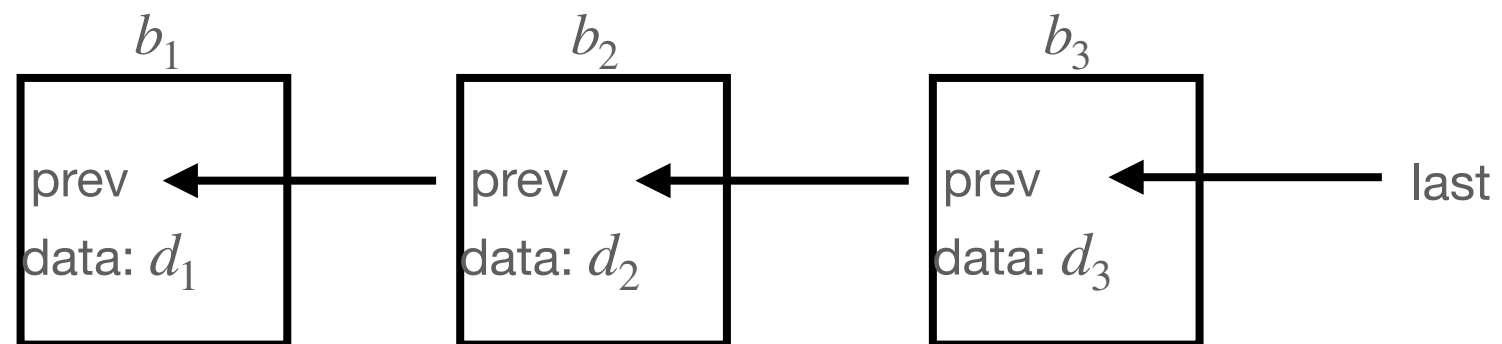
Hash chain

Linked list



```
type Node struct {  
    next pointer  
    data bytes  
}
```

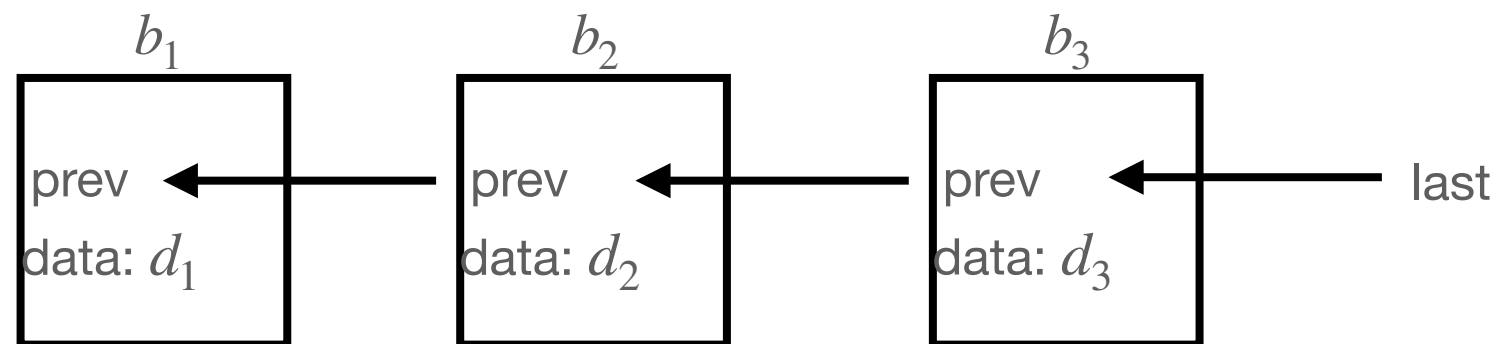
Hash chain



```
type Block struct {  
    prev pointer  
    data bytes  
    prevhash hash  
}
```

- Can hash a block by concatenating fields.
- Blockhash gives id:
 $id_b = H(b.\text{prevhash} || b.\text{data})$
- $b_2.\text{prevhash} = id_{b_1}$

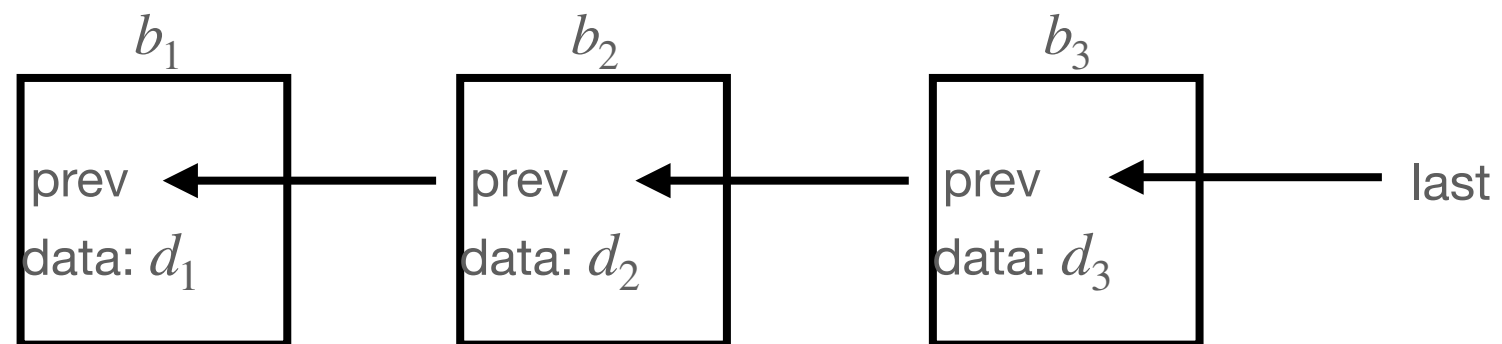
Hash chain



- $id_b = H(b.\text{prevhash} || b.\text{data})$
- Blockchain identified by id_{b_3}
- Changing d_1 changes id_{b_3}
- Removing b_2 changes id_{b_3}
- Adding b'_2 changes id_{b_3}

secured against changes

Hash chain



```
type Block struct {  
    prev pointer  
    data bytes  
    datahash hash  
    prevhash hash  
    timestamp  
}
```

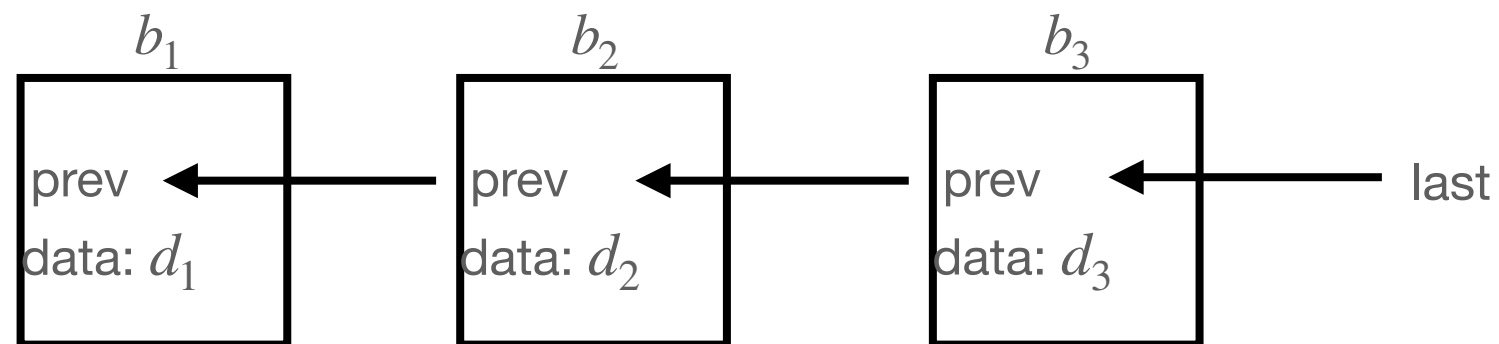
Reduce size:

- $id_b = H(b.\text{prevhash} || b.\text{datahash})$
- $b.\text{datahash} = H(b.\text{data})$

- easy to store
- can proof that data is included

Hash chain

Example: Linked timestamping



Trusted source collects data and publishes a new block, e.g. on newspaper.



Hash chain

Digital Signatures and trusted publishers

$$pk, sk \leftarrow \text{setup}(\kappa)$$

$$\sigma \leftarrow \text{sign}(sk, msg)$$

$$bool \leftarrow \text{verify}(\sigma, msg, pk)$$

Ideas:

- Require a trusted party to sign every new block
- Require m out of n trusted parties to sign a block

Permissioned blockchains!

Merkle trees

Merkle trees

Problem

```
type Block struct {  
    prev pointer  
    data bytes  
    datahash hash  
    prevhash hash  
    timestamp  
}
```

Idea:

- put multiple data items into one block

- Is my item in the block?

Merkle trees

Ideas

Data items: $D_1, D_2, D_3, D_4, \dots$

- $\text{datahash} = h$

Data as a hash

- $h = H(D_1 || D_2 || D_3 || \dots)$

Data as a hashes

- $h = H(D_1) || H(D_2) || H(D_3) || \dots$

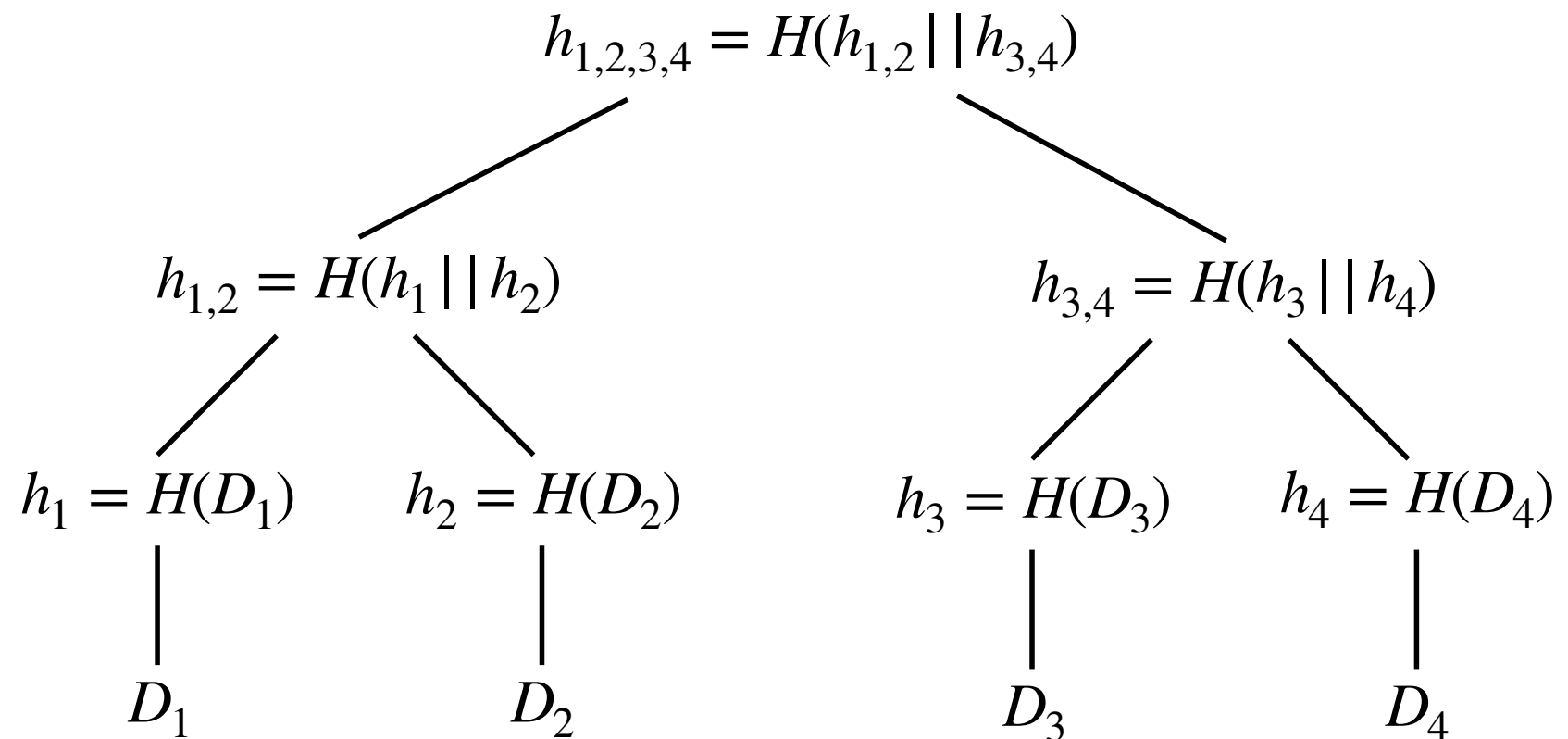
```
type Block struct {  
    prev pointer  
    data bytes  
    datahash hash  
    prevhash hash  
    timestamp  
}
```

Merkle trees

Design

Data items: $D_1, D_2, D_3, D_4, \dots$

- datahash = h

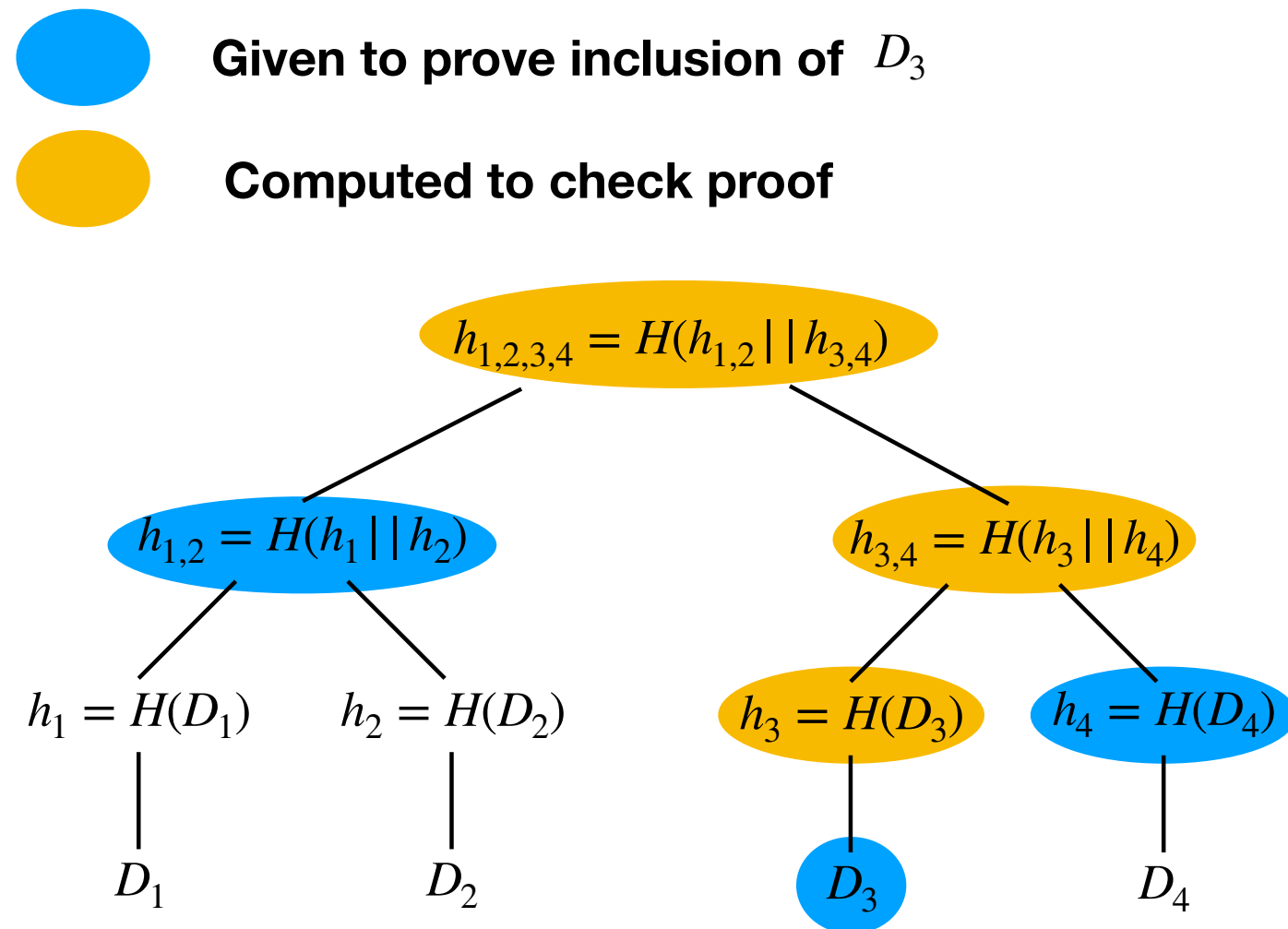


Merkle trees

Proofs

Data items: $D_1, D_2, D_3, D_4, \dots$

- datahash = h



Merkle trees

Proofs

Data items: $D_1, D_2, D_3, D_4, \dots$

- datahash = h

What if we have only 5 data items?

- Duplicate D_5
- Add default element

Signatures and Transactions

Transactions

How can we create an application/cryptocurrency on a blockchain?

- What is in the blocks?
- How to build a meaningful application from it?
- Assume anyone can submit data to the blockchain.

Transactions

Digital Signatures

$$pk, sk \leftarrow \text{setup}(\kappa)$$

$$\sigma \leftarrow \text{sign}(sk, msg)$$

$$bool \leftarrow \text{verify}(\sigma, msg, pk)$$

Ideas:

- Use public key as identity.
- Put signed messages on the blockchain. $\langle msg \rangle_\sigma$
- Signed messages are called *transactions*.

Transactions

Accounts

Transactions are: $\langle pk_{from}, pk_{to}, value \rangle_\sigma$

State is: balance for each public-key

Checks:

- Is signature correct?
- Does pk_{from} have enough money?

Transactions

Accounts

Transactions are: $\langle pk_{from}, pk_{to}, value \rangle_{\sigma}$

Algorithm 1 Account transactions

```
1: balances := [pk]uint
2: for block in chain do
3:   for  $\langle pk_{from}, pk_{to}, value \rangle_{\sigma}$  in block.data do
4:     if !verify(pkto || value, pkfrom,  $\sigma$ ) then
5:       continue
6:     if balances[pkfrom] < value then
7:       continue
8:     balances[pkfrom] − = value
9:     balances[pkto] + = value
```

Transactions

Accounts

Transactions are: $\langle pk_{from}, pk_{to}, value \rangle_\sigma$

State is: balance for each public-key

Checks:

- Is signature correct?
- Does pk_{from} have enough money?

Problems:

- 1. How to deposit money?**
- 2. Replay attack!**

Transactions

Accounts

Transactions are: $\langle pk_{from}, pk_{to}, value \rangle_\sigma$

Deposit:

- Give out some money
- Deposit with someone who has money

Replay attack:

- A signed transaction can be submitted multiple times.
- Sequence numbers!

Transactions

Accounts

Algorithm 2 Account transactions

```
1: balances := [pk]uint
2: sqNrs := [pk]uint
3: for block in chain do
4:   for  $\langle pk_{from}, pk_{to}, value, sqNr \rangle_{\sigma}$  in block.data do
5:     if !verify(pkto || value || sqNr, pkfrom,  $\sigma$ ) then
6:       continue
7:     if balances[pkfrom] < value then
8:       continue
9:     if sqNrs[pkfrom] = sqNr then
10:      balances[pkfrom] − = value
11:      balances[pkto] + = value
12:      sqNrs[pkfrom] ++
```

Idea: do checks when adding transaction to chain.

Transactions

UTXO

UTXO: Unspent transaction output

Idea: *No balances but coins*

- For each coin store pk of owner and unique id
- Transaction spends some coins and creates new ones.

Transactions

UTXO

Transactions:

$$tx = \langle \underbrace{[(id_1, \sigma_1), (id_2, \sigma_2)]}_{\text{Inputs}}, \underbrace{[(pk_a, value_a), (pk_b, value_b)]}_{\text{Outputs}} \rangle$$

State is unspent outputs $map[id](pk, value)$

Transactions

UTXO

Transactions:

$$tx = \langle \underbrace{[(id_1, \sigma_1), (id_2, \sigma_2)]}_{\text{Inputs}}, \underbrace{[(pk_a, value_a), (pk_b, value_b)]}_{\text{Outputs}} \rangle$$

Valid if:

- Inputs refer to unspent outputs.
- Signatures are correct (with outputs public key)
- Value of all inputs larger or equal than all output values.

Transactions

UTXO

Algorithm 3 Transaction validation and maintenance of UTXO

$UTXO := \text{map}[id](value, pk)$

for $tx = \langle inputs, outputs \rangle$ **do**

for $(id, \sigma) \in inputs$ **do**

if $UTXO[id]$ does not exist **then**

abort

▷ invalid transaction

if $\text{verify}(tx, \sigma, UTXO[id].pk) == \text{false}$ **then**

abort

▷ invalid transaction

if sum of values of inputs < sum of values of new outputs **then**

abort

▷ invalid transaction

for $((id, \sigma) \in inputs$ **do**

$\text{remove}(UTXO[id])$

▷ output spent

for $((pk, value) \in outputs$ **do**

$UTXO[newid] = (pk, value)$

▷ add new outputs

Transactions

UTXO

Transactions:

$$tx = \langle \underbrace{[(id_1, \sigma_1), (id_2, \sigma_2)]}_{\text{Inputs}}, \underbrace{[(pk_a, value_a), (pk_b, value_b)]}_{\text{Outputs}} \rangle$$

- No replay attack
- What to sign: $\langle [id_1, id_2], [(pk_a, value_a), (pk_b, value_b)] \rangle$

Transactions

Accounts vs. UTXO

Assuming only valid transactions on chain,
how to verify that a pk has money.

Accounts: Check all received and sent transactions.

UTXO: Check received output and that it is unspent.

Transactions

Accounts vs. UTXO

Assuming only valid transactions on chain,
how to verify that a pk has money.

Accounts: Check all received and sent transactions.

UTXO: Check received output and that it is unspent.

Does UTXO provide anonymity/prevent tracing?

Transactions

Accounts vs. UTXO

Assuming only valid transactions on chain,
how to verify that a pk has money.

Accounts: Check all received and sent transactions.

UTXO: Check received output and that it is unspent.

Does UTXO provide anonymity/prevent tracing?

- Also in UTXO transactions from one sender can be traced.
- But most untracable solutions build on UTXO

Take away

A blockchain is an append only log
secured against changed.

Transactions/state changes are recorded in the blockchain.

Application state can be recreated by applying all transactions.