RAIBLOCKS

Murat Osmanoglu

What is Blockchain?

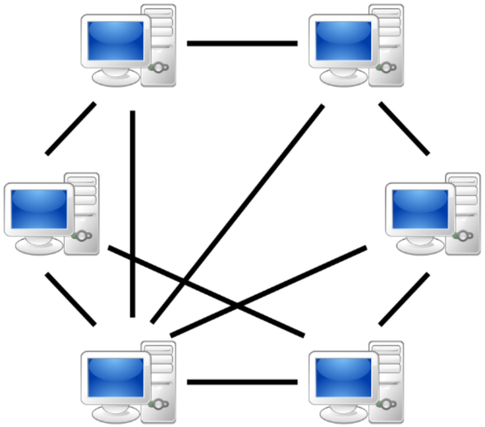
- is used to maintain a continuously growing list of records (transactions).
- enables transactions to be pooled into blocks and recorded
- allows the resulting database(ledger) to be accessed by the different parties
- cryptographically chains blocks in chronological order

What is Blockchain?

someone request a tx

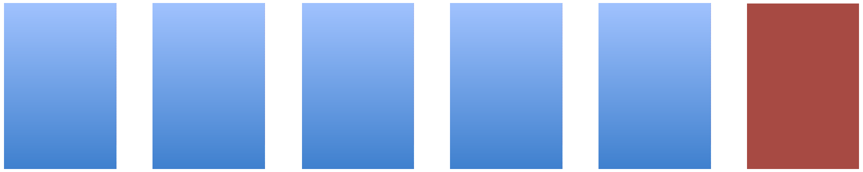
the tx is broadcast
to a P2P network

the content of the transaction is sth
having digital value
(cryptocurrency, contracts, records,
…)



the network of nodes validates
the transaction and the user's status

the transaction combined with other verifed
transaction to create a new block of data.
The new block added to the existing
blockchain



What is Blockchain?

- In blockchain, all parties agree on a specific protocol that determines true state of the database(ledger) at any point in time.

- initiation and broadcasting the transaction (digital signatures with secret-public key pairs)

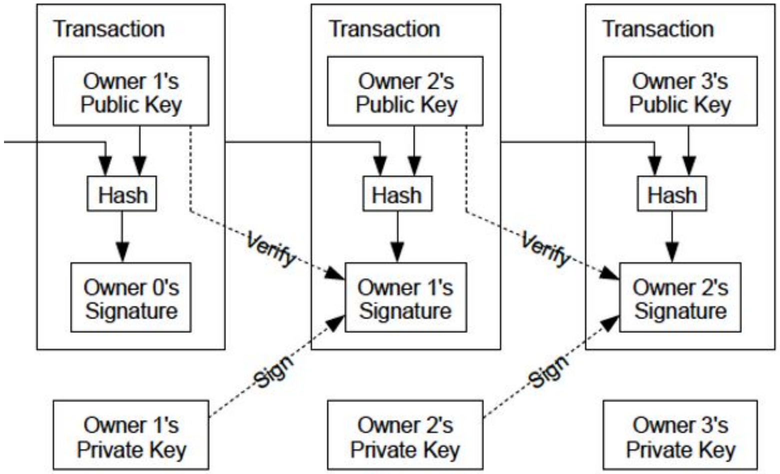
- validation and recording of the transactions (digital signatures and consensus mechanism)

- chaining blocks (hash function)

Raiblocks

- a cryptocurrency with a novel block-lattice architecture where each account has its own blockchain

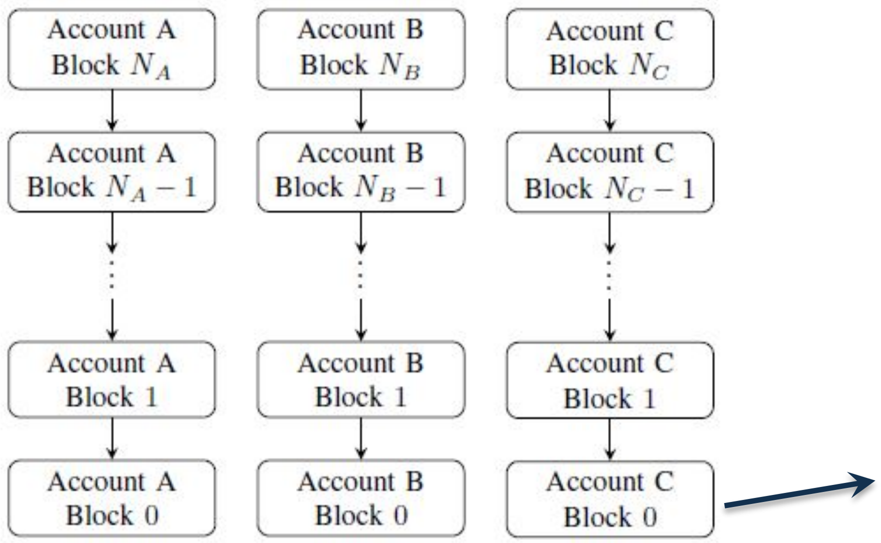
- transactions keep track of account balances rather than transaction amounts (not as in bitcoin)



- it achieves consensus via a balanced-weighted vote on conflicting transaction

Components

- accounts : represented by the public key part of a digital signature key pairs
- blocks : contains a single transaction (can be viewed as the digital encoding of a transaction)
- ledger : global set of accounts where each account has its own transaction chain



open transaction

- nodes : software that runs the RaiBlocks protocol

Model

Genesis Account

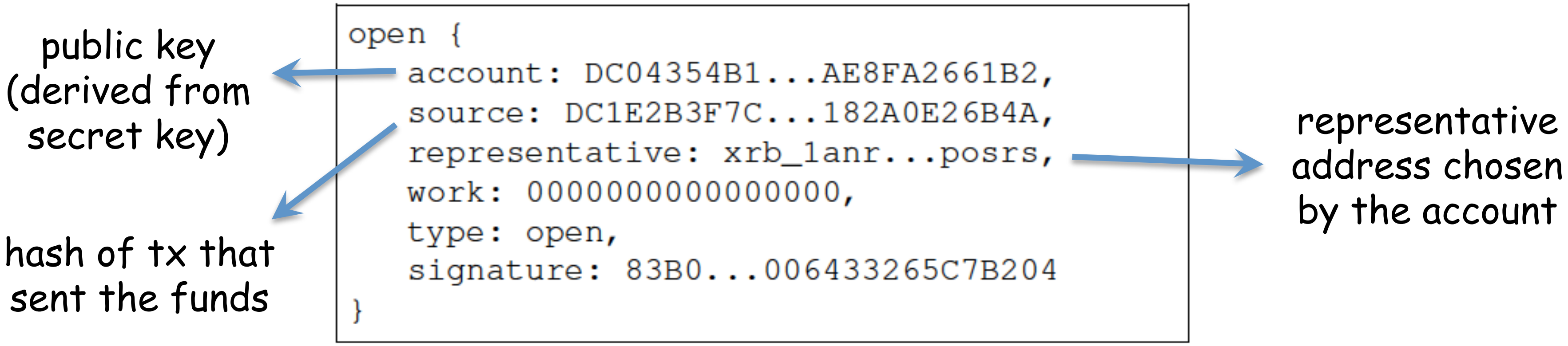
- System initiated with a genesis account that contains the genesis balance (fix amount and never increase)
- The genesis balance is sent to other accounts registered on the genesis-account-chain.

Model

Transactions

- 4 different types of transactions:

  open, send, receive, change



public key (derived from secret key)

hash of tx that sent the funds

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

representative address chosen by the account

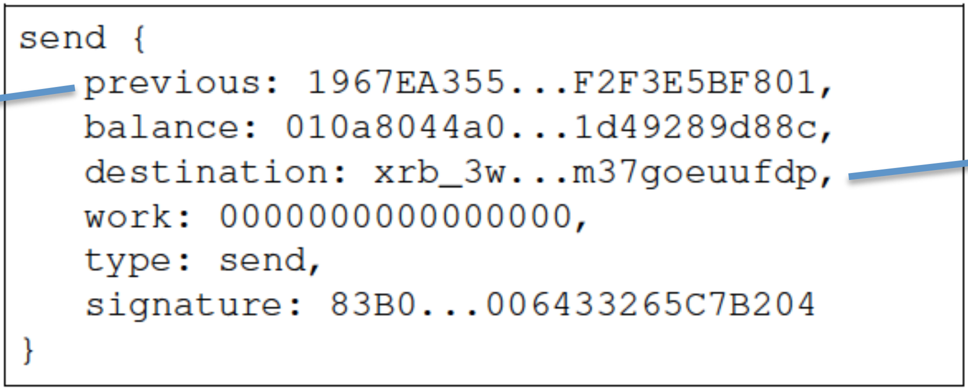
- To create an account, an open transaction should be issued.
- It is the first transaction of every-chain, can be created upon the first receipt of funds

Model

Transactions

- 4 different types of transactions:

  open, send, receive, change



```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

hash of the previous block

the address that the funds will be sent

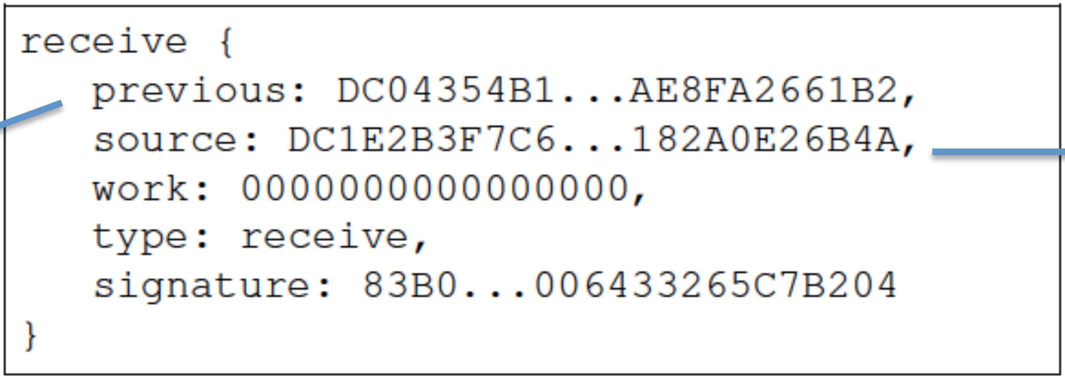
- a send block is immutable once confirmed
- once broadcasted to the network, funds are deducted from the balance of the sender, and wait as pending till the receiver signs a block to accept it (not as bitcoin)

Model

Transactions

- 4 different types of transactions:

  open, send, receive, change



```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

hash of the previous block

hash of the associated send transaction

- once this block is created, the account balance is updated and the funds officially moved to this account

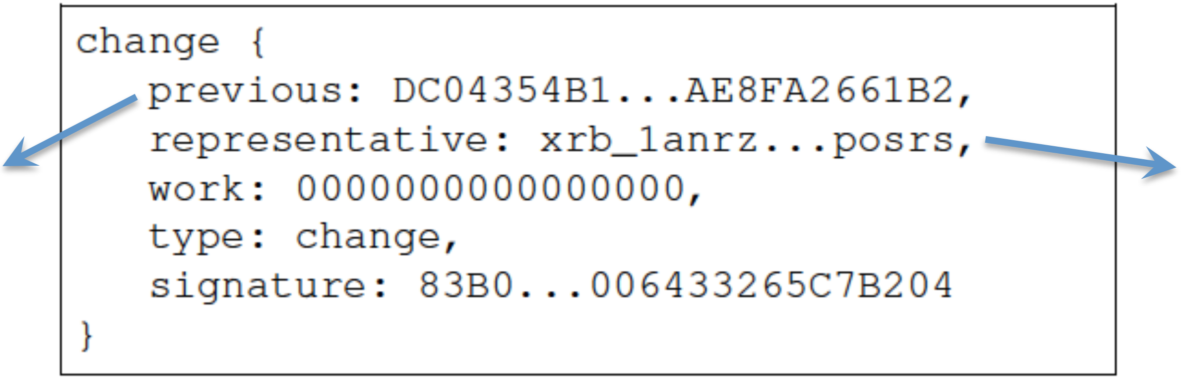
Model

Transactions

- 4 different types of transactions:

  open, send, receive, change

hash of the previous block



```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

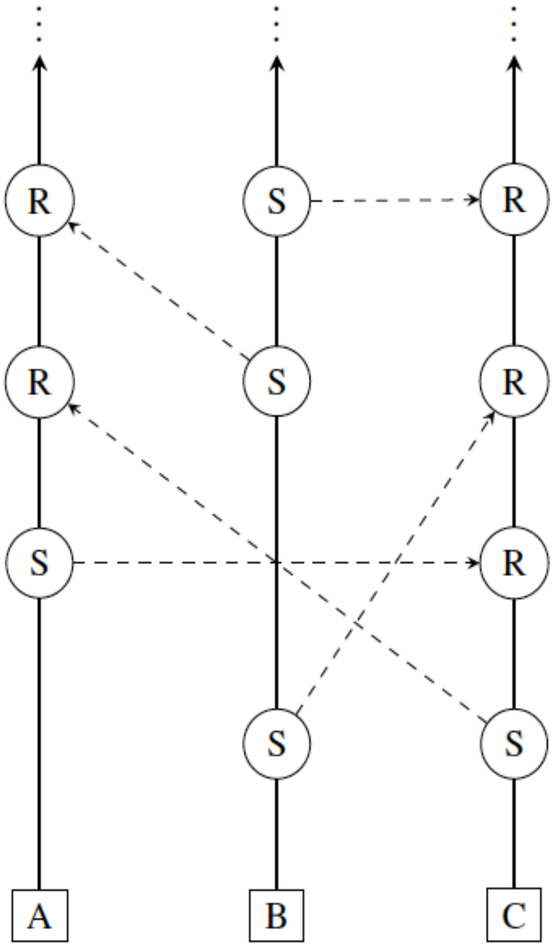
new representative's address

- It changes the representative of an account by subtracting the vote weight from the old representative and adding it to the new one.
- No funds is moved through this transaction

Model

Why two seperate transactions ?

- sequencing incoming transfers that are asynchronous
- keeping transactions small
- isolating settled transactions from unsettled ones
  (settled transactions where an account has created receive blocks)
- the receiving account has control over deciding which transfer arrived first



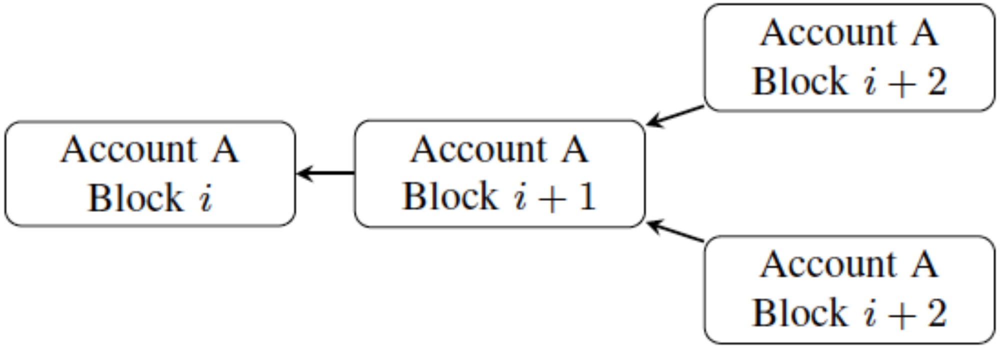
Model

Account Balance

- the account balance is recorded within the ledger
- for a large transfer, assume it has been received from many small transfers, having balance and seperating transactions as send and receive enables to create small fixed size transactions
- some nodes store only latest blocks which have the balance information, still maintain the correctness.

Model

Forks



- After detecting such case, the representative creates a vote referencing the one of the conflicting block in its ledger and broadcast it to the network.

- The nodes observe incoming votes from the other online representatives and decide on the winning block based on the total cumulated weight for that block

Model

Proof of Work

- Every type of transactions has a work field
- It helps the creator to compute a nonce in a way that the hash of the nonce together with the previous field is below a certain threshold
- It is not hard as PoW in bitcoin and it is used as an anti-spam tool
- Once a transaction is sent, the PoW can be precomputed since the previous block field is known.

Model

- Since each account-chain can only be updated by the account's owner, each account-chain can be updated immediately and asynchronously (quick transactions)

- Since nodes only have to record and rebroadcast blocks, hardware requirements are minimal

- A node may either store the entire ledger or a pruned history containing only the last few block of each account's chain (but in setup, it's suggested to verify the entire history, and prune locally)

Attacks

Block Gap Synchronization

- If a node observes a block that does not have the referenced previous block, it might ignore the block (may be some garbage), or might request a resync with another node

- But resync can cause increased amount of traffic in network

- To avoid unnecessary resyncing, nodes will wait until a certain threshold of votes have been observed. If there is not enough votes, the block can be assumed to be junk

Attacks

Transaction Flooding

- A malicious entity could send many unnecessary but valid transactions between accounts under its control to saturate the network (no transaction fees)
- PoW limits the transaction rates the malicious node could generate

Sybil Attack

- An entity could create hundreds of nodes on a single machine to take the control
- Since the voting system is weighted based on the balance, adding extra nodes will not increase powers

Attacks

Penny-Spend Attack

- An entity spends small quantities to a large number of accounts to waste the storage resources of nodes

- PoW limits tis attack and nodes might just store last blocks to maintain the correctness

Precomputed PoW

- An entity may create a number of sequential blocks with their PoWs. Then at a certain point, it performs a Denial of Service by flooding the network with a lot of valid transactions

- It can cause short damage, and transaction-rate limiting can be applied (they are working on that)