Blockchain

Nicoleta Preda

January 8, 2019

Agenda

Hash Functions

A hash function:

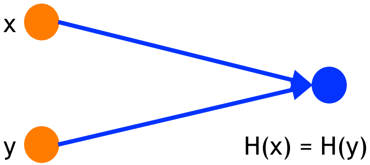
- takes any string as input
- fixed-size output (e.g., Bitcoin uses 256 bits)
- efficiently computable

Security properties:

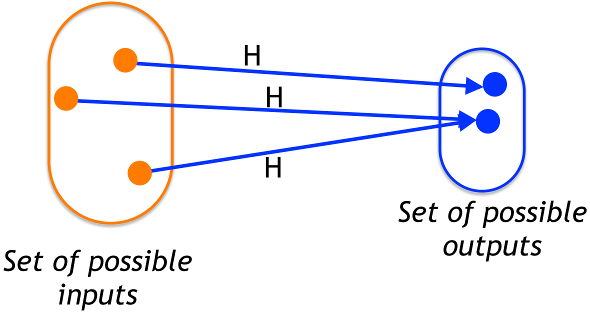
- collision-free
- hiding
- puzzle-friendly

Hash Property 1: Collision-Free

Nobody can find x and y such that
x != y and H(x)=H(y)



Collisions do exist ...



. . . but can anyone find them?

How to find a collision?

try 2^{130} randomly chosen inputs

99.8% chance that two of them will collide

This works no matter what H is . . .
. . . but it takes too long to matter

Is there a faster way to find collisions?

- For some possible H's, yes.(Example ?)
- For others, we don't know of one.

No H has been proven collision-free.

Application: Hash as message digest.

Useful to verify if two very large files are identical.

If we know $H(x) = H(y)$,
it's safe to assume that $x = y$.

To recognise a file that we saw before,
just remember its hash.

Useful because the hash is small.

Hash Property 2: Hiding

We want something like this:
Given H(x), it is infeasible to find x.

Hash Property 2: Hiding

We want something like this:
Given H(x), it is infeasible to find x.

Exercise:

H("heads")

H("tails")

How easy is to find x here?

Hiding Property:

If r is chosen from a probability distribution that has high min-entropy, then given $H(r \mid x)$, it is infeasible to find x.

High min-entropy means that the distribution is "very spread out", so that no particular value is chosen with more than negligible probability.

Application: Commitment

- Want to "seal a value in an envelope", and "open the envelope" later.
- Commit to a value, reveal it later.

Commitment API:

- To seal msg in envelope:
  select key, a random 256-bit value,
  commit(msg) $\rightarrow$ ($\underline{\text{com=H(key} \mid \text{msg)}}$, H(key)), then publish com
- To open envelope:
  publish key, msg
  anyone can check validity: H(key | msg) == com

Security Properties for the Commitment API

Hiding:	Given the result of H(key | msg), infeasible to find msg.
Binding:	Infeasible to find msg != msg' such that H(key | msg) == H(key | msg')

Hash Property 3: Puzzle-Friendly

Puzzle-friendly:

For every possible output value y, if k is chosen from a distribution with high min-entropy, then it is infeasible to find x such that $H(k \mid x) = y$.

Application: Search Puzzle

Puzzle-friendly property implies that no solving strategy is much better than trying random values of x.

Given a "puzzle ID" id (from high min-entropy distrib.),
and a target set Y:
Try to find a "solution" x such that
$H(id \mid x) \in Y$.

Agenda

Crypto Principles (from Eduard Felten's tutorial)

- Hash Functions
- Hash Pointers and Data Structures
- Digital Signatures
- Public Keys as Identities
- Simple Cryptocurrencies

Distributed Architecture

Bitcoin

Conclusion

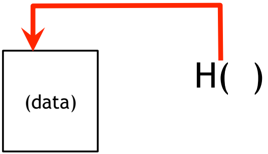
Principle

hash pointer is:

- pointer to where some info is stored, and
- hash of the info (cryptographic hash)

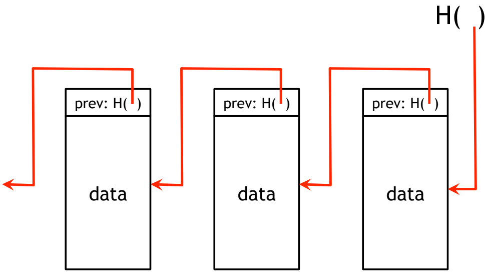
if we have a hash pointer, we can

- ask to get the info back, and
- verify that it hasn't changed



Distributed Data Structure: Block Chain

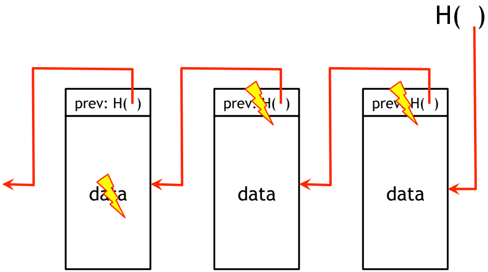
linked list with hash pointers = "block chain"



Use-case: tamper-evident log

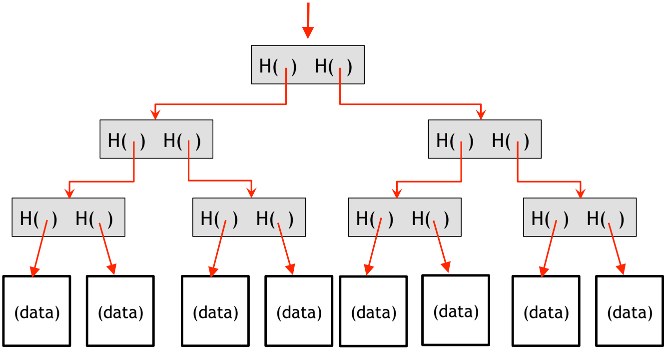
Data is added to the end.
No one can change what has been previously added.

Detecting Tampering



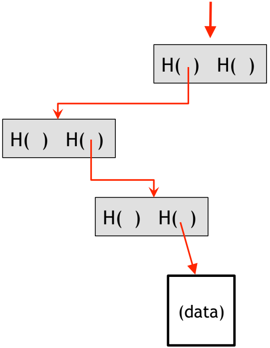
Merkle Tree

binary tree with hash pointers = "Merkle tree"



Proving Membership in a Merkle Tree

show O(log n) items



Advantages of Merkle Trees

- Just need to remember the root hash
- Can verify membership in O(log n) time/space

Variant: sorted Merkle tree

- can verify non-membership in O(log n)
- show items before, after the missing one

Agenda

Digital Signatures

- Only you can sign, but anyone can verify
- Signature is tied to a particular document
  can't be cut-and-pasted to another doc

API for digital signatures

- generateKeys(keysize) $\rightarrow$ (sk, pk)
  sk: secret signing key
  pk: public verification key
- sign(sk, message) $\rightarrow$ sig
- verify(pk, message, sig) $\rightarrow$ isValid

Bitcoin uses ECDSA standard (Elliptic Curve Digital Signature Algorithm)

Requirements for Digital Signatures

"valid signatures verify":

verify(pk, message, sign(sk, message)) == true

"can't forge signatures":

adversary who:

knows pk

gets to see signatures on messages of his choice

can't produce a verifiable signature on another message

Practical Constrains of Digital Signature Algortihms

- Algorithms are randomized
  $\rightarrow$ need a good source of randomness
- Have an upper bound limit on the size of the message
  $\rightarrow$ fix: sign the result of the hash function

Agenda

Crypto Principles (from Eduard Felten's tutorial)

- Hash Functions
- Hash Pointers and Data Structures
- Digital Signatures
- Public Keys as Identities
- Simple Cryptocurrencies

Distributed Architecture

Bitcoin

Conclusion

public key = an identity (= address in Bitcoin)

Decentralised identity management

- to "speak for" pk, you must know matching secret key sk
- anybody can make a new identity at any time
  make as many as you want!
- no central point of coordination

Privacy

- Addresses not directly connected to real-world identity.
- But observers can link together an address's activity over time, make inferences.

Agenda

Two Simple Cryptocurrencies

- GoofyCoin 
- ScroogeCoin 

GoofyCoin

- Goofy can create new coins (by design).
- The new coin belongs to Goofy

signed by sk_{Goofy}
CreateCoin [uniqueCoinID]

GoofyCoin (continue)

- A coin's owner can spend it.
- Goofy pays Alice.
- Alice owns the coin now.

signed by sk_{Goofy}

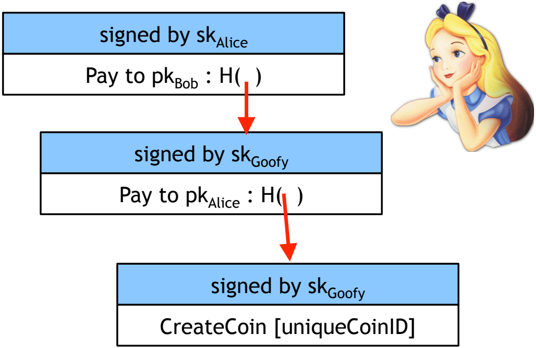
Pay to pk_{Alice} : H( )

signed by sk_{Goofy}

CreateCoin [uniqueCoinID]

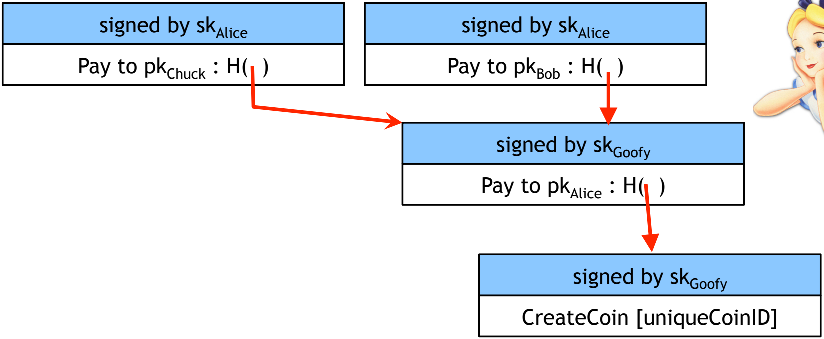
GoofyCoin (continue)

- The recipient (Alice) can pass on the coin again.
- Bob owns the coin now.



GoofyCoin Issue: Double-Spending Attack

- Alice passes the same coin to Bob and Chuck.
- Which of the two blocks is valid?

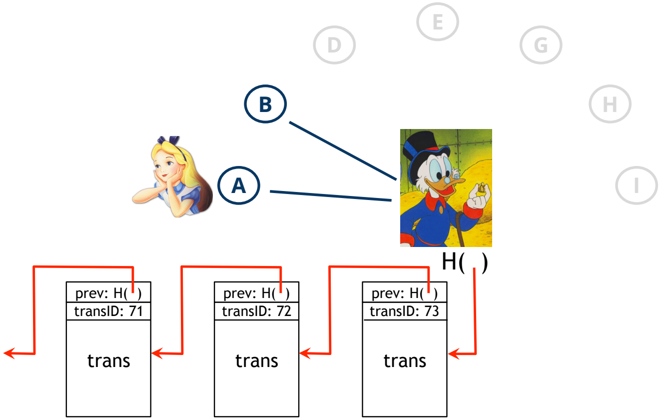


Double-Spending: One of the main design challenges in digital currencies!

- Centralised Solution 
- Decentralised Solution (Bitcoin)

Scrooge Coin: A Centralised Solution

- A central bank (Scrooge) maintains a history of all transactions.
- History of all transactions: a block chain, signed by Scrooge



Detail

- We consider one transaction per block-chain.
- In reality, for optimisation, a block contains multiple transactions.

Example Transaction

- CreateCoins Transaction creates new coins.
- Is valid because Scrooge, the central authority, decides so.
- Everyone trusts Scrooge.

transID: 73 type:CreateCoins

coins created:

num	value	recipient	
0	3.2	0x...	← coinID 73(0)
1	1.4	0x...	← coinID 73(1)
2	7.1	0x...	← coinID 73(2)

Example Transaction

- CreateCoins Transaction creates new coins.
- Is valid because Scrooge, the central authority, decides so.
- Everyone trusts Scrooge.

transID: 73 type:CreateCoins

coins created:

num	value	recipient	
0	3.2	0x...	← coinID 73(0)
1	1.4	0x...	← coinID 73(1)
2	7.1	0x...	← coinID 73(2)

Example Transaction (continue)

- PayCoins transaction consumes (and destroys) some coins, and creates new coins of the same total value
- The transaction is valid if:
  - consumed coins valid, not already consumed,
  - total value out = total value in, and
  - signed by owners of all consumed coins

transID: 73	type:PayCoins	
consumed coinIDs: 68(1), 42(0), 72(3)		
coins created:		
num	value	recipient
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...
Signatures owners of consumed		

Coins are Immutable

How to subdivide a coins?

- Create a new PayCoins transaction.
- Pay out two new coins to yourself.

- Issue with the centralises solution:

  You need to trust Scrooge (the central bank).
- Solution:

  Distributed, self-organising peer-to-peer solution.

  No central control, no unique point of failure.

Agenda

Crypto Principles (from Eduard Felten's tutorial)

Distributed Architecture

- P2P Overlay Networks
- Disemination Algorithms
- Security
- Consistency

Bitcoin

Conclusion

Peer-to-Peer (P2P) Networks

Advantages

- Robustness: the disconnection of a peer is without consequences.
- Responsability and costs are shared among peers.

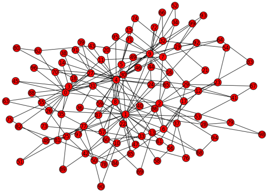
Disadvantages

- Complex coordination as peers come and leave (response time).
- Malicious participants may take control over the network (security).
- No control of the dissemination of information (privacy).

P2P Topology

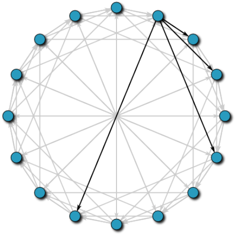
Unstructured

- Random connections between peers
- No overhead for maintaining the structure



Structured: Distributed Hash Tables (DHT)

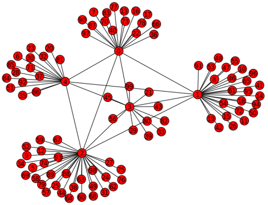
- Ring of peers (hashed ids)
- Every peer assigned specific links
- Superimposed binary look-up trees
- Cached links for fast look-up
- High overhead when peers churn



P2P Topology

Unstructured but not random

- Emergent Super-Peers
- Possible self-organization: peers connect “well connected” peers
- Some overhead caused by super-peers



Blockchain: Challenges and Approaches

- Q: Which topology is more suited for a blockchain?

Unstructured P2P: Join the Network

Challenge: no central entity to contact

Bitcoin's Approach:

- Use pre-configured IP addresses.
- Rely on volunteers to host DNS servers.
- Then, learn IP addresses of other nodes.
- A peer connects to a certain number of peers (default: 8)
- Reciprocally, a peer accepts incoming connections (on average 30).
- Periodically ping peers to check liveness
- Swap so-so nodes for better ones (several hours).

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?
- A: Every peer stores the history of the transactions.

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?
- A: Every peer stores the history of the transactions.

- Q: How are new transactions handled?
- Example: “Alice wants to pay Bob 5 cents."

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?
- A: Every peer stores the history of the transactions.

- Q: How are new transactions handled?
- Example: “Alice wants to pay Bob 5 cents."
- A: A p2p algorithm is used to “broadcast” the message

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?
- A: Every peer stores the history of the transactions.

- Q: How are new transactions handled?
- Example: "Alice wants to pay Bob 5 cents."
- A: A p2p algorithm is used to "broadcast" the message

- Q: What happens when a peer receives two identical messages?

Blockchain: Challenges and Approaches

- Q: Where is the ledger stored?
- A: Every peer stores the history of the transactions.

- Q: How are new transactions handled?
- Example: “Alice wants to pay Bob 5 cents."
- A: A p2p algorithm is used to “broadcast” the message

- Q: What happens when a peer receives two identical messages?
- A: Unique identifiers are necessary!

Agenda

Disemination Algorithms

Problem: broadcast a message to all the peers.

Solutions:

- Flooding
- Gossip

Flooding Algorithms

- Each peer forwards the new information to its connections.
- To reduce bandwidth consumption
  peers can first echange hashes of messages,
  and transfer only those unseen before.

Gossip Algorithms

Principle:

Gossiping is the endless process of
randomly choosing two members
and subsequently letting these two exchange information
(Kermarrec/Van Steen, “Gossiping in distributed systems”)

Gossip Algorithm: Rumor Mongering

1. Peers are initially ignorant.
2. When a peer learns a new message it becomes a hot rumor.
3. Periodically, the chooses a random peer
   and sends (pushes) the rumor to it.
4. Eventually, the peer loses interest in spreading the rumor.

Gossip Algorithm: Anti-Entropy

1. Each peer p periodically contacts a random peer q.
2. p and q engage information:
   updates from p are transferred to q (push),
   or vice-versa (pull),
   or in both direction (push-pull).

Rumor Propagation

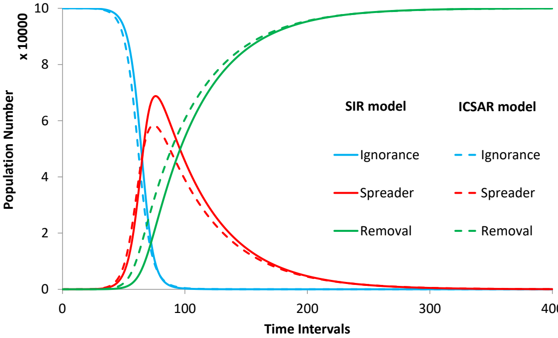


Figure from Zhang & all: Dynamic 8-state ICSAR rumor propagation model considering official rumor refutation.

Peer Sampling in Gossiping Algorithms

- Each peer periodically exchanges information with a set of peers.
- The choice of this set is crucial.
- Ideally, it should be a uniform random sample of all online peers.

Peer Sampling in Gossiping Algorithms

- Each peer periodically exchanges information with a set of peers.
- The choice of this set is crucial.
- Ideally, it should be a uniform random sample of all online peers.

Issue:

- Infeasible to maintain a list with all peers in the P2P overlay.

Peer Sampling: A Gossip-like Solution

- Every peer maintains a relatively small list of peers.
- It periodically modifies it using a gossiping procedure:
  it selects a peer
  and the two echange (swap) parts of their lists

Overview Disemination Algorithms

Problem: broadcast a message to all the peers.

Solutions:

- Flooding: robust,
  less efficient (O(n2))
- Gossip: robust,
  efficient (O(nlogn)),
  relative high latency,
  only probabilistic guarantees for reliability

Agenda

Crypto Principles (from Eduard Felten's tutorial)

Distributed Architecture

- P2P Overlay Networks
- Disemination Algorithms
- Security
- Consistency

Bitcoin

Conclusion

Blockchain: Challenges and Approaches

- Q: How to make sure Alice does not “double spends”?

Blockchain: Challenges and Approaches

- Q: How to make sure Alice does not “double spends”?
- Every peer verifies the transaction before adding it to the ledger.

Blockchain: Challenges and Approaches

- Q: How to make sure Alice does not "double spends"?
- Every peer verifies the transaction before adding it to the ledger.
- Q: What can get wrong?

Blockchain: Challenges and Approaches

- Q: How to make sure Alice does not “double spends”?
- Every peer verifies the transaction before adding it to the ledger.
- Q: What can get wrong?
- Nota bene: decentralized p2p network

Sybil Attack

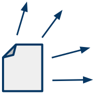
- Alice sets up multiple identities

Sybil Attack

- Alice sets up multiple identities
- Creating a new peer requires:
  memory/disk space and
  processor time.
- Several peer can co-exist on the same computer.
- The majority of the peers belong to Alice.

Solution: Proof of Work

Every peer:



1.	Collect and verify submitted transactions to form a block
2.	Proof of Work: Solve a puzzle (competition between peers)
3.	The winner can append its block to the chain Hence, it announces the block to the network

Step “Proof of Work" as a competition

- Q: Why?

Agenda

Information Propagation

Issue: Network becomes inconsistent once a new block is generated

Case of study Bitcoin 2013

- Average time till a node receives a new block : 12.6 seconds
- Longtail: 5% of nodes do not have the new block after 40seconds
- Consequence: blockchain forks
- 169 forks were observed during a period of 10,000 generated blocks

Decker and Wattenhofer IEEE P2P'13

Agenda

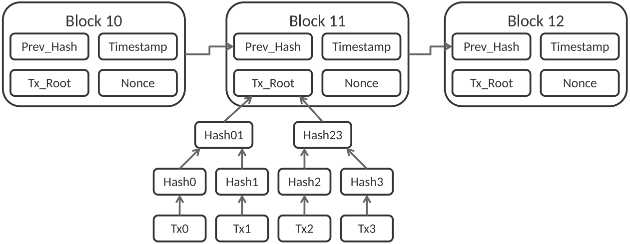
Bitcoin

- The block chain provides Bitcoin's public ledger.

  ledger = the principal book recording the commercial transactions

The Blockchain

- A block is an ordered and timestamped record of transactions.
- Each full node in the Bitcoin network independently stores a block chain.



Agenda

Crypto Principles (from Eduard Felten's tutorial)

Distributed Architecture

Bitcoin

- Data Structures
- Nakamoto Consensus
- Proof-of-Work
- Consensus
- Issues

Conclusion

Bitcoin: “Nakamoto Consensus"

- New transactions are broadcasted to all peers (miner).
- Each miner collects[1] new transactions into a block.
- In each round a random[2] miner gets the privilege add its block to the chain. To do so, the miner broadcasts the block in the network.
- Other miners accept the block if all the transactions in it are valid.
- Miners express their acceptance of the block by including its hash in the next block they create.

[1] Some systems prioritise transactions those fees are higher (a better reward).
[2] Actually, the winner of the proof-of-work puzzle.

Miners

- Miners are peers that:
  receive and validate transactions,
  group them into a block
- Miners race to solve a puzzle (poof-of-work) in order to publish their[3] block
- Miners invest a large amount of computer in the race.
- The reward is valid only if other nodes accept the published block.

[3] Create/earn a bitcoin reward for each block published ("mined")

Agenda

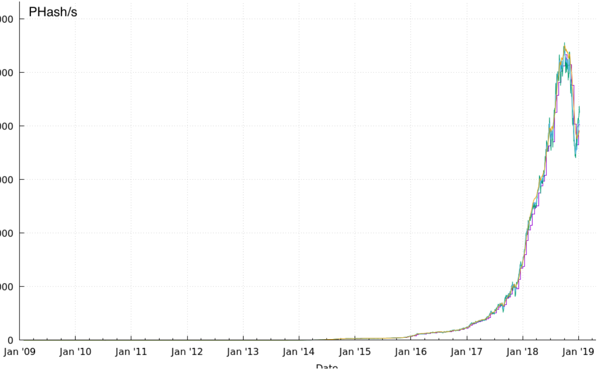
Proof-of-Work: The Hash Lottery

- Difficult Problem (Proof of Work) for the miner:
  finding a number (known as a nonce) that
  when nonce is added to a given input data,
  the value of the hash begins with a number of zeros:
  $H(\text{nonce} \mid \text{data}) = 0000000xxxx$
- In the Bitcoin world this is what "mining" is.
- A miner's win probability is proportional to its compute power.
  In this way, the next miner to publish is selected randomly.

Why is this problem important?

Proof-of-Work in Bitcoin:

- Effect: a lot of hash-power is spent on guessing winning lottery numbers that satisfy the difficulty of the problem.
- Incentive: “Guess" numbers in order to obtain the reward from the network.



As of January 2018, the global hashrate is about 17 exahash per second (EH/s).

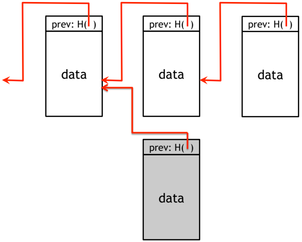
The Payout

- The node that finds the best solution to the challenge is granted a reward.
- Originally in Bitcoin it was 50 new coins.
- If a nodes receives two competing solutions it selects the one that has:
  - higher number of transactions included in the block,
  - higher number of zeros.

Agenda

Arriving at Consensus

How to avoid forks in the block chain?



- When a miner links to a block, it accepts as the head of the valid chain.
- If others disagree, then the miner's block is worthless.
  ... so miners have an incentive to get it right!
- If a block A is posted "too late" (e.g., the block in grey) or is invalid,
  other miners ignore it, and
  build the chain in another direction.
- The longest chain wins: defines the "next puzzle"

Transaction Confirmation

- Transaction accepted in a candidate block ⇒ the transaction is valid.
- Confirmation: Every new block accepted into the chain after the block containing the transaction counts as a confirmation.
- Coins are not considered mature until there have been 6 confirmations.
  - means 1h if a 10 minute block cadence
- New mined coins are not valid until about 120 confirmations.
  - assures that a node with more than 51% of the total hash-power does not pull off fraudulent transactions
- Checkpoints to insure that every to prevent complete history rollback.

Agenda

Node with Limited Functionality

Leave out some functionality

- Calculation of proofs of work (mining)
- Store only block headers instead of complete blocks
  $\rightarrow$ incomplete verification of transactions
  $\rightarrow$ trust others (verification of the transaction + proof of work)
- Acceptance of incoming connections

Node with Limited Functionality

Leave out some functionality

- Calculation of proofs of work (mining)
- Store only block headers instead of complete blocks
  $\rightarrow$ incomplete verification of transactions
  $\rightarrow$ trust others (verification of the transaction + proof of work)
- Acceptance of incoming connections

Potential attacks on SPV

- Attacker controls victim's internet connection.
- It double-spends and computes own proof(s) of work.
- Victim does not check for double-spending.

Privacy issues

Transaction graph

- Link different Bitcoin addresses of a user

Peer-to-Peer Network

- Find origin (IP address) that broadcasted a transaction:
  create fake identities and let many peers, and connect to you
  connect to peers that accept incoming connections
  as described in a talk by Dan Kaminsky, CC Congress 2011

Scalability

Broadcast

- If Bitcoin would process VISA's 2,000 transactions per second,
  $\rightarrow$ this would lead to a traffic of 8 MB/s

Denial-of-Service Attacks

- Goal: overload or stop the transmission of transactions/blocks
- Hard to counteract in a P2P overlay network
- Limit block size

Conclusion Distributed Consensus

Distributed consensus can allow:

- Distrustful parties to maintain clean state
- Completely unambiguous rules about validity
- Removing authentication and identity as essential

Other applications to Blockchain Technology

- Registeries
- Authoritative Systems of Record
- Directory Services
- Timestamping Services (“Proof of Existence”)
- Counter-party Exchanges