Introduction to Blockchains

CMPS 4760/6760: Distributed Systems

Cryptocurrencies

- A **cryptocurrency** (or **crypto currency**) is a digital asset designed to work as a medium of exchange that uses strong cryptography to secure financial transactions, control the creation of additional units, and verify the transfer of assets

--- Wikipedia

Cryptocurrencies

- Prevent people from
  - tampering with the state of the system
  - equivocation: e.g., If Alice convinces Bob that she paid him a digital coin, she should not be able to convince Carol that she paid her that same coin.
- Lack of central authority
  - Cryptography
  - Decentralized control

Overview

- From Cryptography to Cryptocurrency
- Repeated Consensus and Blockchain
- Bitcoin and Nakamoto's Blockchain Protocol

- References:
  - A. Narayanan, et al., "Bitcoin and Cryptocurrency Technologies", 2016
  - E. Shi, "Foundations of Distributed Consensus and Blockchains", 2021

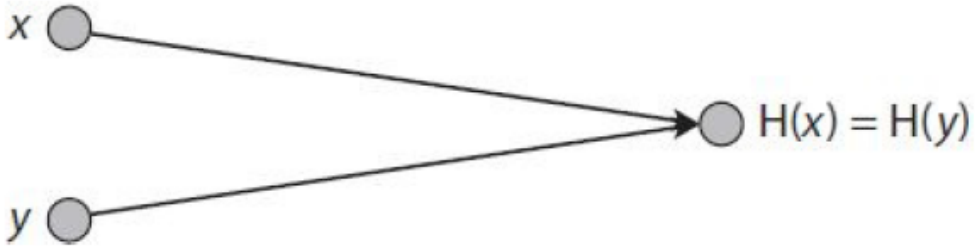
Cryptographic hash functions

- A hash function is a math function with three properties
  - input can be any size
  - a fixed-size output, e.g., 256 bit
  - efficiently computable

- Cryptographic hash functions need to satisfy
  - Collision resistance
  - Hiding
  - Puzzle friendliness

Collision Resistance

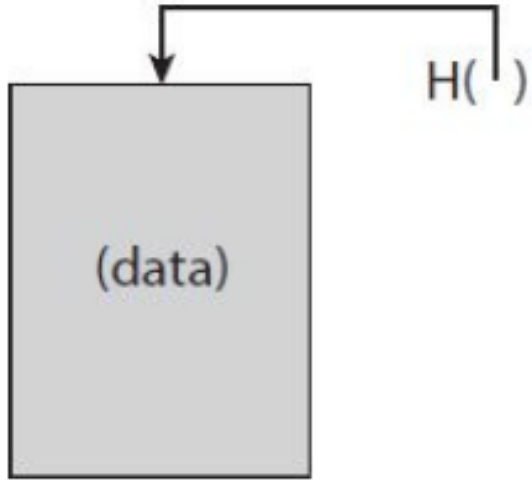
- A hash function H is collision resistant if it is infeasible to find two values, x and y, such that $x \neq y$, yet $H(x) = H(y)$.



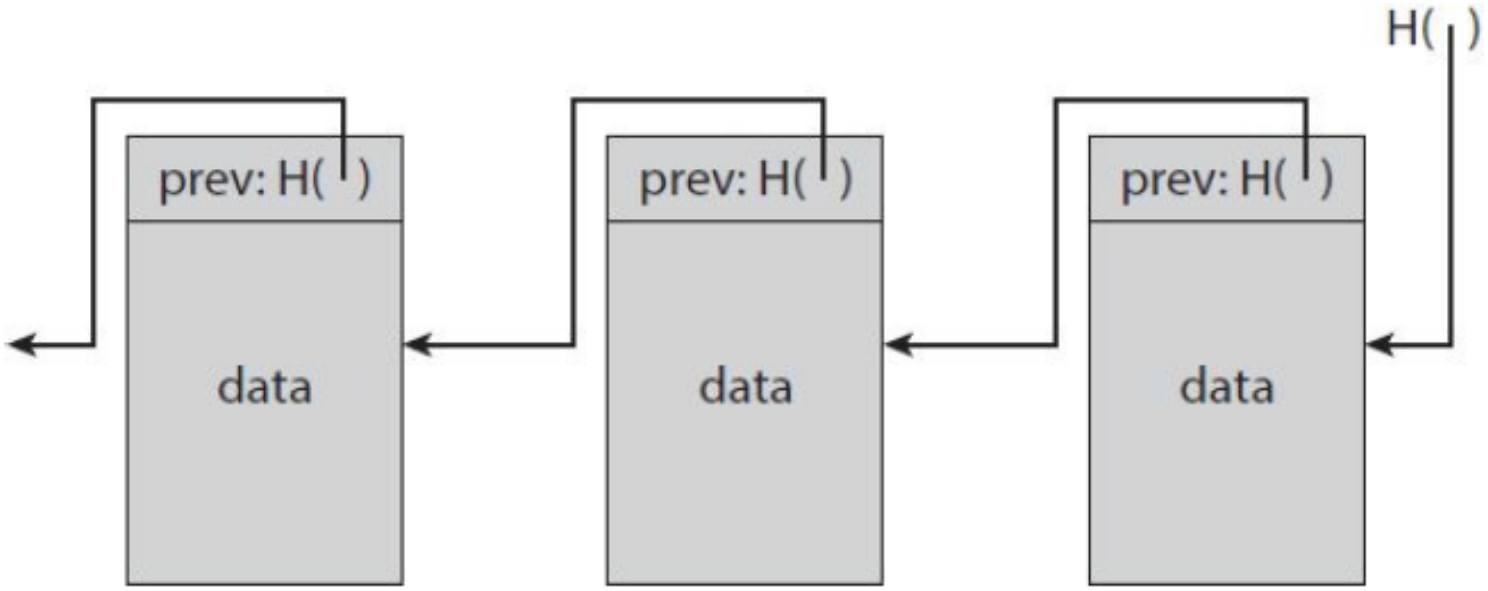
- Application: message digests
- e.g., Alice uploads a really large file, and she wants to be able to verify later that the file she downloads is the same as the one she uploaded.
  - Just needs to keep the hash of the original file.
  - computes the hash of the downloaded file and compares it to the one she stored.

Hash pointers

- A hash pointer is a pointer to where data is stored together with a cryptographic hash of the value of this data at some fixed point in time.



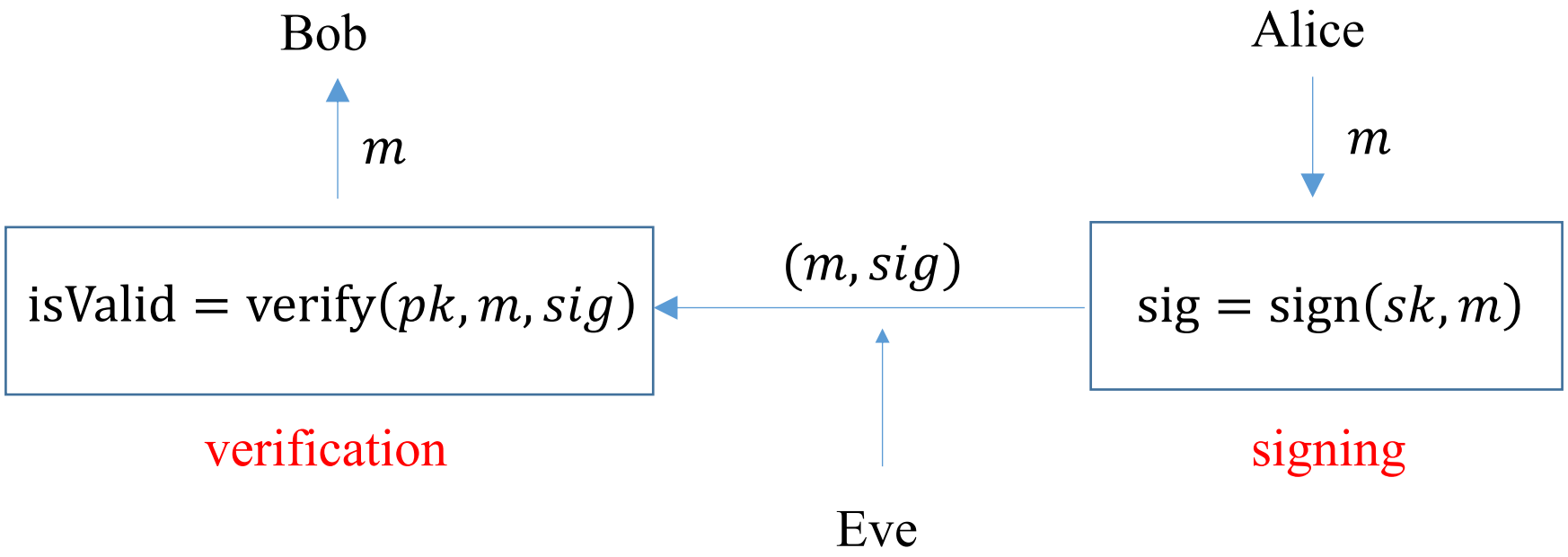
Blockchains



Use case: Tamper-evident log

- an adversary modifying data anywhere in the blockchain will result in the hash pointer in the following block being incorrect.
- If we store the head of the list, then even if an adversary modifies all pointers to be consistent with the modified data, the head pointer will be incorrect, and we can detect the tampering.

Digital Signatures



- Alice has a key pair (sk, pk) = generateKeys (*keysize*)
- Valid signatures must verify: $\text{verify}(pk, m, \text{sign}(sk, m)) = true$
- Unforgeability: it's computationally infeasible to forge signatures

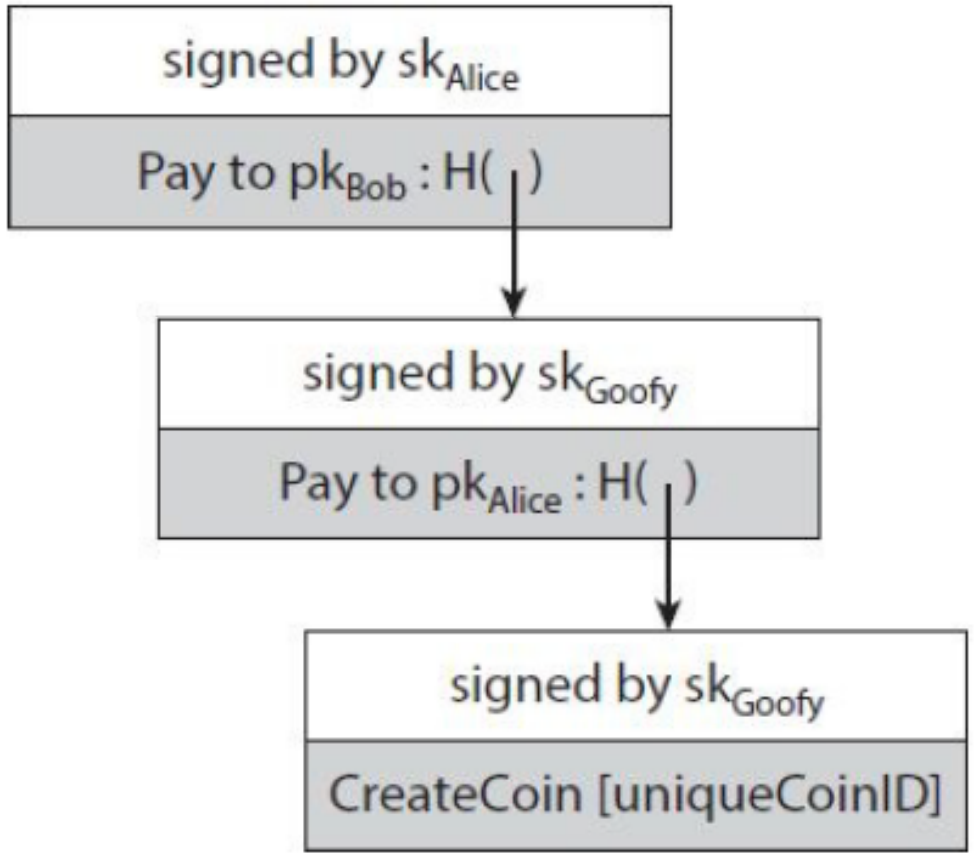
Digital Signatures

- Public keys as identities
  - or hashes of public keys as identities (called “addresses” in Bitcoin)
  - enables decentralized identity management
  - privacy is still a big concern
    - Over time, the identity that you create makes a series of statements.
    - People see these statements and thus know that whoever owns this identity has done a certain series of actions.
    - They can start to connect the dots, using this series of actions to make inferences about your real-world identity

Goofycoin

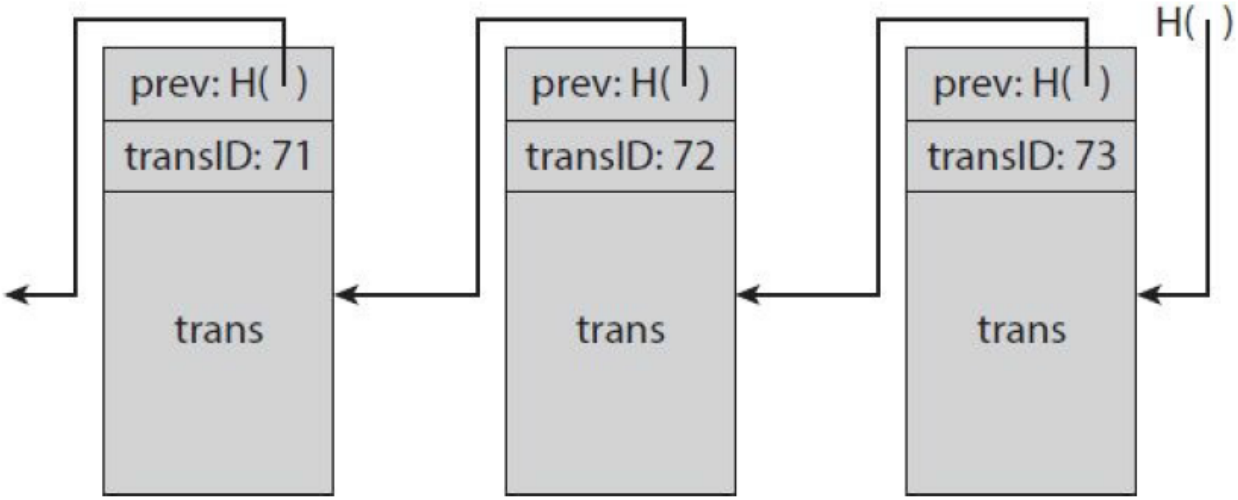
- Goofy can create new coins by simply signing a statement that he's making a new coin with a unique coin ID.
- Whoever owns a coin can pass it on to someone else by signing a statement that says, "Pass on this coin to X" (where X is specified as a public key).
- Anyone can verify the validity of a coin by following the chain of hash pointers back to its creation by Goofy, verifying all signatures along the way.

- Suffering from double-spending attack



Scroogecoin

- Scrooge publishes an append-only ledger containing the history of all transactions
  - Implemented as a blockchain
  - Each block has one transaction
  - Scrooge digitally signs the final hash pointer



Scroogecoin - Two kinds of transactions

A CreateCoins transaction is valid if it is signed by Scrooge

transID: 73	type:CreateCoins	
coins created		
num	*value*	*recipient*
0	3.2	0x…
1	1.4	0x…
2	7.1	0x…

coinID 73(0)
coinID 73(1)
coinID 73(2)

A PayCoins transaction is valid if

- consumed coins are valid and signed by owners
- consumed coins have not been consumed before (no double-spending)
- total values coming out equals to total values that went in

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		

Problem with Scroogecoin

- Scroogecoin prevents double spending
- Scrooge cannot forge transactions
- But Scrooge has too much influence
  - It could stop endorsing transactions from some users unless they transfer some mandated transaction fee to him.
  - It can create as many new coins for himself as he wants.
  - It could get bored of the whole system and stop updating the blockchain
- Problem: centralization

What we want

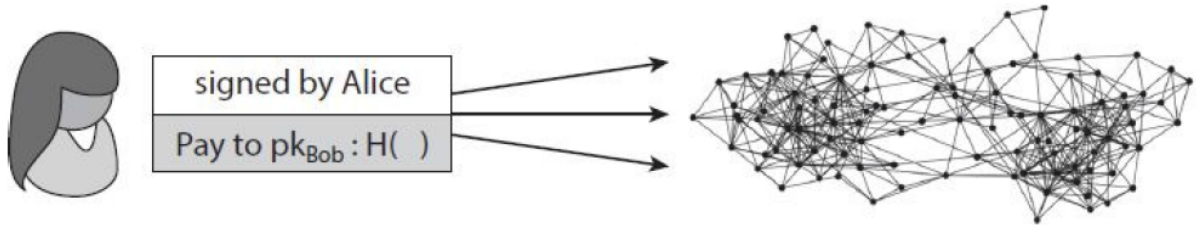
- A decentralized system where
  - All users need to agree on a single published blockchain as the authoritative history of all transactions.
  - They must all agree on which transactions are valid, and which transactions have actually occurred.
  - They also need to be able to assign IDs in a decentralized way.
  - Finally, the minting of new coins also needs to be decentralized.

Overview

- From Cryptography to Cryptocurrency
- Repeated Consensus and Blockchain
- Bitcoin and Nakamoto's Blockchain Protocol

Consensus in Bitcoin

- Bitcoin is a peer-to-peer system: anybody can run a Bitcoin node



- When Alice wants to pay Bob, she broadcasts a transaction to all nodes
  - Running a node is not necessary for Bob to receive the funds

Consensus in Bitcoin

- At any point in time, all nodes have
  - A sequence of blocks each containing a list of transactions that they have reached consensus on
  - a pool of outstanding transactions (may differ from each other)
- Consensus problem: what's the next block to be included in the blockchain

Repeated Consensus

- So far in our lectures, we have considered single-shot consensus
- Practical applications often require reaching consensus repeatedly over time
- In Bitcoin, a distributed set of nodes maintain an ever-growing public ledger which records the sequence of all transactions that have taken place so far
- Blockchain is an abstraction for repeated consensus
  - classically called state machine replication in distributed systems literature

Assumptions

- A system with n nodes (processes)
  - Each receives transactions from an external environment
  - transactions are represented as bit strings that possibly need to abide by certain validity rules
  - The nodes each maintain a growing linearly ordered log of transactions.
- Synchronous system
  - Nodes communicate in rounds
  - Honest nodes' messages can take at most Δ rounds to be delivered to an honest recipient

Requirements of a Blockchain

- LOG_i^r: node i's finalized log in round t
- Consistency: for any honest nodes i and j, and for any round numbers t and r, it must be that either LOG_i^r is a prefix of LOG_j^t, or LOG_j^t is a prefix of LOG_i^r
- T_{conf}-liveness: If an honest node receives some transaction tx as input in some round r, then by the end of round $r + T_{conf}$, all honest nodes' local logs must include tx.
  - T_{conf} is called confirmation time and is a function of the number of nodes n and the maximum network delay Δ

Blockchain from Sequential Composition of Byzantine Broadcast

- Byzantine Broadcast (BB): A multi-valued extension of Byzantine Generals Problem
  - Either using signatures or not
- n nodes are numbered $0, 1, \ldots, n-1$
- Each BB instance runs in R number of rounds

- In every round kR $(k = 0, 1, 2 \ldots)$, spawn a new BB protocol with $L_k := (k \bmod n)$ as the designated sender
- L_k collects transactions it has received as input, but that have not been included in its current log, and inputs the concatenation of all such transactions into BB_k
- At any time, a node's output log is defined as the concatenation of the output of the BB protocols that have finished.

Blockchain from Sequential Composition of Byzantine Broadcast

Theorem: Suppose that the BB protocol adopted realizes Multi-Valued Byzantine Broadcast for a network of n nodes and tolerating up to f corruptions, then the above blockchain construction satisfies consistency and $O(Rn)$-liveness also for the same n and f, where R denotes the round complexity of BB.

- Problems with the protocol
  - Long confirmation time
  - Byzantine Broadcast requires a permissioned system
    - Oral messages: pairwise authenticated
    - Signed messages

Overview

- From Cryptography to Cryptocurrency
- Repeated Consensus and Blockchain
- Bitcoin and Nakamoto's Blockchain Protocol

Consensus in Bitcoin

- Permissionless
  - Anyone is free to join the consensus protocol at any time
  - No a-priori knowledge of the identities of the participants
  - Unauthenticated channels: anyone can impersonate anyone else
  - Leading to Sybil Attack

Proof-of-Work (PoW)

- Players need to solve computational puzzles to cast votes (called mining)
- Roughly, a player's voting power is proportional to its computational power
- Moreover, the blockchain protocol guarantees consistency and liveness as long as the majority of the mining power in the system is honest

Nakamoto's Blockchain

- Each honest node maintains a blockchain, denoted chain, at any point of time.
- The first block $\mathsf{chain}[0]$ is a canonical block called the genesis
- Every other block has the format $\mathsf{chain}[i] := (h_{-1}, \eta, \mathrm{txs}, h)$, where
  - h_{-1}: hash of the previous block
  - η: a puzzle solution
  - txs: a set of transactions to be confirmed
  - h: hash of the present block

Mining

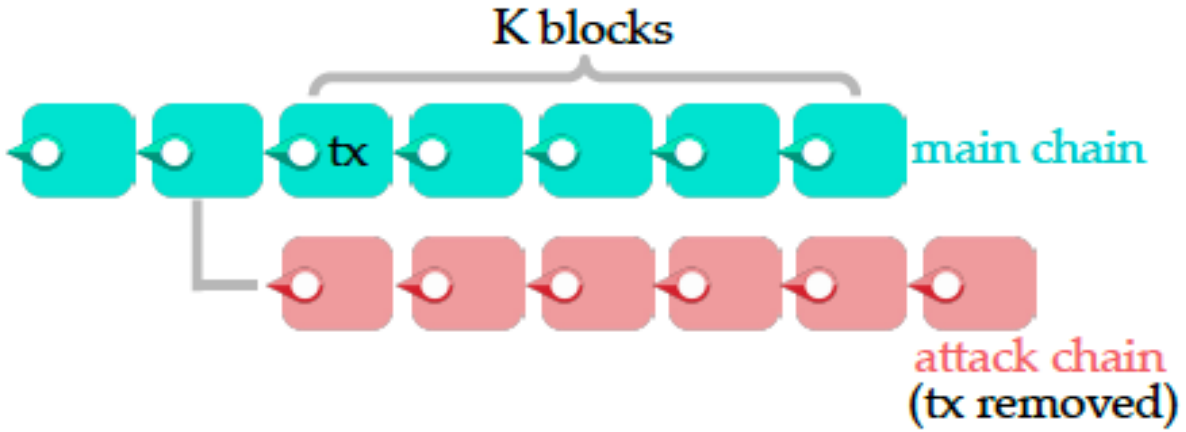
- Given a blockchain chain with last block $(\_, \_, \_, h_{-1})$
- To "mine" a new block, a miner would try random puzzle solutions $\eta \in \{0,1\}^{\lambda}$ and check if

$$H(h_{-1}, \eta, \text{txs}) < D_p$$

  - $H: \{0,1\}^* \to \{0,1\}^{\lambda}$: a Proof-of-Work (PoW) oracle, implemented as a hash function
  - D_p: a difficulty parameter
  - In Bitcoin, D_p is chosen such that in expectation it takes all miners combined 10 minutes to mine a new block
  - If η solves the puzzle, $\big(h_{-1}, \eta, \text{txs}, H(h_{-1}, \eta, \text{txs})\big)$ forms a valid block extending from chain

Longest Chain

- Miners always try to mine a block off the longest chain it has seen
- At any time, all but the last K blocks in the longest chain are considered final.
- The larger the K, the more confident we are about tx's finality.



- To undo tx, the adversary would have to mine an attack fork off some prefix of chain$[:-K]$, and the attack fork must be longer than the main chain for it to win

Nakamoto's Blockchain (simplified)

- Nodes that are newly spawned start with initial chain containing only a special genesis block: chain $:= (0,0,\perp,H(0,0,\perp))$
- Whenever a node hears a fresh message from the network or receives a new transaction as input, it echoes the message or transaction to everyone else
- In every round, a node tries to mine a new block off the longest chain seen so far
  - Pick a random solution solution $\eta \in \{0,1\}^{\lambda}$ and issue query $h = H(h_{-1}, \eta, \text{txs})$
  - If $h < D_p$, append the newly mined block to chain and send chain$||(h_{-1}, \eta, \text{txs}, h)$ to everyone
- Whenever a node hears a message chain' from the network, if chain' is a valid and is longer than its current local chain, replace chain by chain'
- At any time, a node's finalized log is defined to be chain$[:-K]$, i.e., the longest chain observed so far removing the last K blocks.

Choosing the Mining Difficulty Parameter

- In Bitcoin, the difficulty parameter is chosen such that on average, all miners combined take 10 minutes to mine the next block
- In practice, many consider $K = 6$ to be secure enough — this means it could easily take an hour for a transaction to confirm!
- However, the puzzles' difficulty cannot be arbitrarily lowered
  - As doing so could break the consistency of the consensus protocol

Choosing the Mining Difficulty Parameter

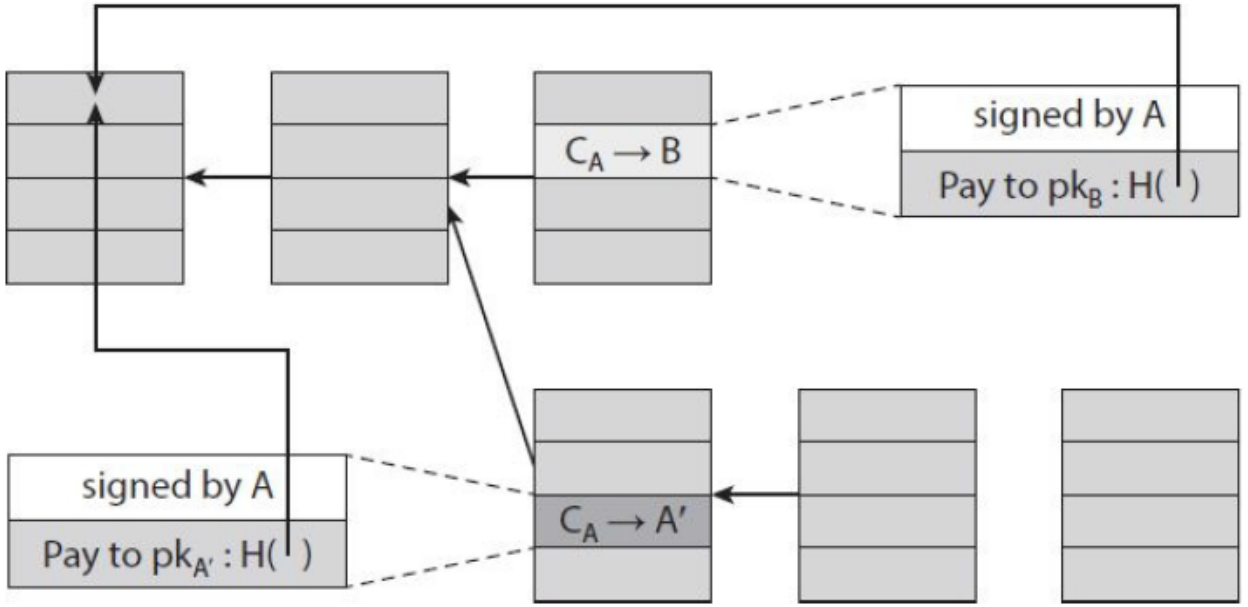
- Choose D_p such that the probability that any player mines a block in a round is $p \in (0, 1)$
- Suppose that there are n nodes, 51% of which is honest
- The probability that the honest nodes combined can mine a block in a round is roughly $1 - (1 - p)^{0.51n} \approx 0.51pn \ll 1$
- The expected number of rounds till a new honest block is mined is roughly $\frac{1}{0.51pn}$
- It takes Δ rounds to propagate the block to the honest nodes, while the adversary may not need to suffer from the same Δ delay
- Then roughly speaking, every $\frac{1}{0.51pn}$ rounds, we end up wasting Δ rounds
- The discounted honest mining power can be defined as $\frac{\frac{1}{0.51pn}}{\frac{1}{0.51pn}+\Delta} 0.51n \approx (1 - 0.51pn\Delta)0.51\mathrm{n}$
- To ensure consensus, we require that the honest mining power, even when discounted by the network delay Δ, must exceed the corrupt mining power!

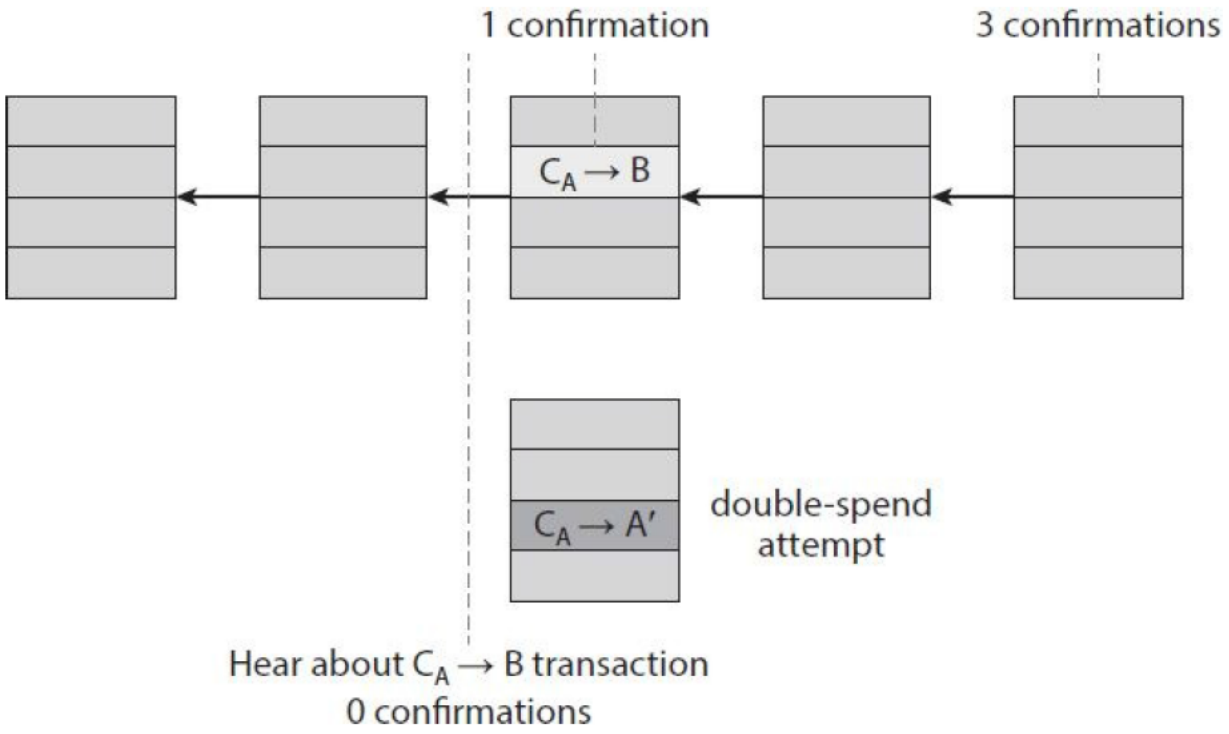
Properties of Nakamoto's Blockchain

- With $1 - \mathrm{negl}(\lambda)$ probability over the choice of the randomized execution of Nakamoto's blockchain, the consensus property is satisfied.
  - $\lambda -$ the security parameter
  - $\mathrm{negl}(\lambda)\ -$ for any fixed polynomial function $p(\lambda)$, there exists λ_0 such that for any $\lambda > \lambda_0$, $\mathrm{negl}(\lambda) < 1/p(\lambda)$
- With $1 - \mathrm{negl}(\lambda)$ probability over the choice of the randomized execution of Nakamoto's blockchain, the liveness property is satisfied with confirmation time bounded by $\Theta\left(\frac{K}{\alpha} + \Delta\right)$
  - α is the expected number of blocks mined by honest nodes in each round.

A double-spend attempt

- Alice creates two transactions: one in which she sends Bob bitcoins, and a second in which she double spends those bitcoins by sending them to a different address, which she controls





- Bob should wait to release the merchandise until the transaction with which Alice pays him is included in the blockchain and has several confirmations.

Recap

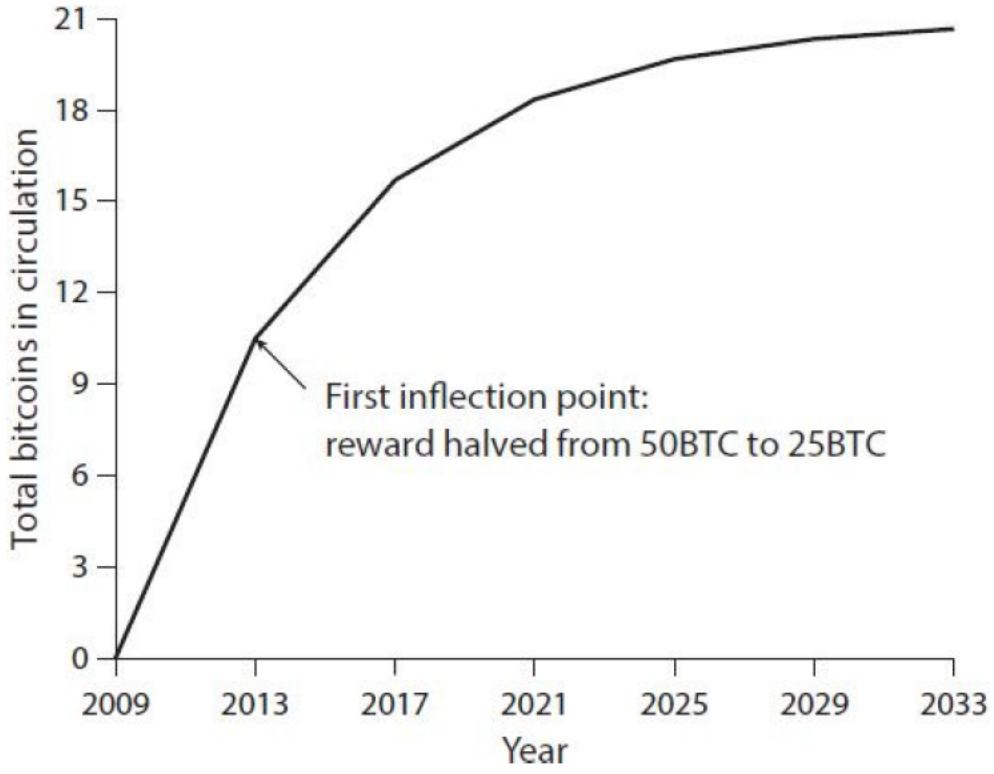
- protection against invalid transactions is entirely cryptographic, but is enforced by consensus
  - a cryptographically invalid transaction won't end up in the long-term consensus chain because a majority of the nodes are honest and won't include an invalid transaction in the block chain
- protection against double spending is purely by consensus
  - the double-spend probability decreases exponentially with the number of confirmations if the majority of nodes are honest

Incentives

- Can we punish nodes with double-spend attempts?
  - not really, hard to detect
  - besides, nodes don’t have identities
- Instead, we can reward nodes that created the blocks that did end up on the long-term consensus chain

Block Reward

- The node that creates a block gets a reward if the block ends up on the long-term consensus branch
- The block reward is 25 bitcoins as of 2015 and halves with every 210,000 blocks created (~every 4 years)
  - This is the only way in which new bitcoins can be created.
  - Limit the total supply of bitcoins to 21 million
  - This block reward will run out in 2140



Transaction Fees

- The creator of any transaction can choose to make the total value of the transaction outputs less than the total value of its inputs

- Whoever creates the block that first puts that transaction into the block chain gets to collect the difference, which acts a transaction fee.

Selfish Mining

- Since miners make some money for mining each block, a selfish miner should want to mine as many blocks as possible
- The Nakamoto's consensus protocol is vulnerable to selfish-mining attack

(a) A selfish miner mines a block **B** and withholds it.

(b) When an honest miner mines an equal-length block **B′**, the selfish miner immediately releases **B** and races with **B′**.

Selfish Mining

- Assume network's delay $\Delta = 0$, the selfish miner controls $\rho < 0.5$ fraction of the mining power.
- Consider a very long window in which a total of T blocks are mined.
- Number of blocks mined by the selfish miner $\approx \rho T$, whereas the number of blocks mined by honest nodes $\approx (1-\rho)T$.
- Every block the selfish miner mines can erase one block mined by honest nodes.
- In the final blockchain, there would only be $(1-\rho)T - \rho T = (1-2\rho)T$ blocks mined by honest nodes.
- The fraction of blocks controlled by the selfish miner is $1 - \frac{1-2\rho}{1-\rho}$
  - $1 - \frac{1-2\rho}{1-\rho} \approx 0.96$ if $\rho = 0.49$