6.857 Recitation 03: Bitcoin

Conner Fromknecht

February 19, 2016

Today

- Transactions
- Double Spending Problem
- Proof of Work
- Blockchain
- Security

Transactions

The primitive building block of cryptocurrencies is the *transaction*, which records the transfer of ownership of something from a sender to a receiver.

- Bitcoin stores value in *addresses*, which are calculated as the hash of the receiver's public key, $H(PK)$
- A transaction is a digitally signed statement that approves the transfer funds between addresses. Typically includes sender's address $H(PK_0)$, receiver's address $H(PK_1)$, amount, and a signature from SK_0 on the prior fields.
- Each transaction is digitally signed, forming a chain of signed transfers from all previous owners

Double Spending Problem

Problem: Need to prevent spending the same coin twice. Since information is easily copied, spending a coin does not remove possession from the sender, and can potentially be reused.

Solution: Timestamp Server
Establishes time-ordering to transactions, on a first come first serve basis.

- Publish the hash of many of transactions at regular intervals
- Proves data existed at certain point in time $\Rightarrow$ partial ordering of transactions
- Hashes are chained, each addition commits all prior hashes

Centralization makes this trivial, but how do we make a *distributed* timestamp server?

Proof of Work

Cryptographic puzzles used to prove *expenditure* of a resource. Typically a large asymmetry between solving and verifying—should be difficult to solve and trivial to verify.

Bitcoin PoW: Find n such that $H(data, n) < 2^{256-D}$, i.e. find a hash with D leading 0's

- $Pr[H(data, i) < 2^{256-D}] = 1/2^D$—only one D–bit prefix starts with D 0's
- Solving requires $\Theta(2^D)$ hashing attempts, using $O(1)$ memory
- Verifying requires computing one hash
- Difficulty self-adjusts, computed using moving average over past two weeks

Proof of Work *rate limits* the submission of valid requests. Allows network to synchronize between subsequent writes.

Blockchains

Composed of two data structures, blocks and block headers.

- *Block:* A list of transactions.
- *Block Header:* (Notable fields provided)
  - Hash of previous block header, $H(Header_{t-1})$
  - Merkle Root over block, one transaction per leaf
  - Nonce, n

Blockchains provide an append only database.

1. All chains extend the *genesis block*, the hardcoded root block.
2. Miners append to chain by computing Proofs of Work on block headers.
3. Miners are rewarded by including a *coinbase transaction*, which pays a predetermined amount to an address of the miner's choosing. This is how new Bitcoin are distributed.
4. A block is valid if the PoW satisfies D and the transactions are valid.
5. PoW commits previous block header and a Merkle Root over new transactions. Changing an entry invalidates that block's and future blocks' Proofs of Work with high probability.
6. Hash function requires Collision Resistance and Pseudorandomness.

Informal Mining Algorithm
Given the previous header $Header_{t-1}$:

1. Gather list of unprocessed transactions
2. Compute Merkle Root of transactions r
3. Choose nonce $n \xleftarrow{\$} \mathbb{Z}$

4. Assemble $Header_t = (H(Header_{t-1}), r, n)$
5. If $H(Header_t) < 2^{256-D}$, broadcast to peers and restart mining algorithm with $Header_t$
6. Increment n, go to step 4

Security

Forks in the blockchain may occur naturally due to the time distribution of solved blocks and/or network propagation. We are primarily concerned with an attacker's ability to intentionally divert his computing resources and mine a sidechain whose length outpaces the honest chain, thus becoming the new main chain. This is exactly the nature of a double spending attack, where a previously confirmed transaction can be reverted by a malicious miner.

Sketch of proof from Bitcoin whitepaper [3]. Define the following:

- p — probability that an honest node finds the next block
- q — probability that a malicious miner finds the next block
- q_z — probability that a malicious miner will overtake the main chain from z blocks behind

The race to extend the honest and malicious chain can be modeled as a Binomial Random Walk. We find that

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}.$$

- $Pr[\text{PoW success}] = 1/2^D$
- Uniform chance of success $\Rightarrow$ p and q are related to hash rate (# of guesses per second)
- Therefore, secure against double spending if *majority* ($p > q$) of hashing power is honest.

Bitcoin's security requires that $p > q$, often interpreted "at least 51% of the hashing power is honest". This property ensures that the attacker's success probability drops exponentially in z, the depth of the fork. For Bitcoin, this implies that the longer the receiver waits to acknowledge a transaction, the more protected they are from double spending. It is recommended that users wait at least 6 confirmations before accepting a transaction.

A Bitcoin textbook was recently published by Princeton, which is available for free [1]. If you're interested in more formal models of blockchain security, this paper analyzes the underlying consensus protocol [2].

References

[1] e. a. Arvind Narayanan. Bitcoin and cryptocurrency technologies. `https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf`, 2016.

[2] N. L. Juan A. Garay, Aggelos Kiayias. The bitcoin backbone protocol. `https://eprint.iacr.org/2014/765.pdf`, 2014.

[3] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.